МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
(МИНПРОСВЕЩЕНИЯ РОССИИ)

Департамент государственной политики в сфере воспитания, дополнительного образования и детского отдыха

Каретный Ряд, д. 2, Москва, 127006
Тел. (495) 587-01-10, доб. 3400
Факс (495) 587-01-13
E-mail: d06@edu.gov.ru

29.10.2020 № 06-1188

Руководителям органов государственной власти субъектов Российской Федерации, осуществляющих государственное управление в сфере образования

О проведении Олимпиады

Департамент государственной политики в сфере воспитания, дополнительного образования и детского отдыха Минпросвещения России информирует о проведении ежегодной Межрегиональной олимпиады школьников имени И.Я. Верченко (далее-Олимпиада), организаторами которой являются Академия ФСБ России, Академия криптографии Российской Федерации и Федеральное учебно-методическое объединение в системе высшего образования «Информационная безопасность».

Олимпиада проходит по двум направлениям: математика и криптография, информатика и компьютерная безопасность.

К участию в Олимпиаде приглашаются школьники 8 - 11 классов.

Подробная информация для школьников о сроках, порядке участия и местах проведения Олимпиады размещена на сайте Академии http://academy.fsb.ru и на сайте Олимпиады http://v-olymp.ru.

Олимпиада проводится в два этапа.

Отборочный этап олимпиады *по математике и криптографии* проводится на сайте Олимпиад *http://v-olvrnp.ru* с 1 по 22 ноября 2020 года в дистанционном формате. Заключительный этап олимпиады - 29 ноября 2019 года в очном формате.

Отборочный этап олимпиады *по информатике и компьютерной безопасности* проводится на сайте Олимпиад *http://v-olvmp.ru* с 1 декабря 2020 года по 24 января 2021 года в дистанционном формате. Заключительный этап олимпиады - 31 января 2021 года в очном формате.

Прошу проинформировать образовательные организации вашего региона о проведении Олимпиады.

Дополнительную информацию по вопросам участия в Олимпиаде можно получить по телефонам *(926) 393 7730, (495) 989 3745, представитель оргкомитета - Катышев Сергей Юрьевич.*

Приложение: в электронном виде

Заместитель Директора департамента

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 60982534537CE13E236850D257510EDC02620A34
Владелец **Колударова Ольга Павловна**
Действителен с 30.09.2019 по 30.12.2020

О.П. Колударова

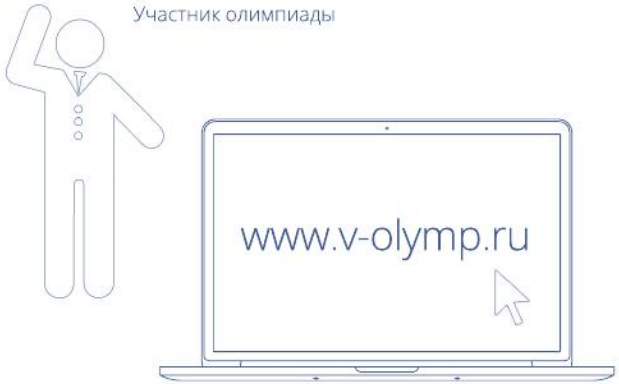
Сереброва Е.В.
(495) 587-01-10, доб. 3095

Как участвовать в олимпиадах?

Олимпиада проходит в несколько этапов,

1. **Отборочный этап**
   Проходит на сайте www.v-olymp.ru
2. **Заключительный этап**
   Проходит в очном формате
3. **Закрытие олимпиады**
   Подведение итогов олимпиады. Награждение призеров.

Возможности участников при регистрации на v-olymp.ru



На сайте каждый участник может

Протестировать свои знания
Ознакомительный этап

Подготовиться к олимпиаде
Курсы дистанционного обучения

Победителям и призерам олимпиады, предоставляются льготы при

поступлении
в Академию ФСБ России

Контактная информация

Заключительные этапы олимпиад
проходят по адресу:

Остановка общественного транспорта

Контактные телефоны

(495) 989 34 22

Адреса сайтов олимпиад в сети-интернет

www.v-olymp.ru

Ваши вопросы по электронной почте

support@v-olymp.ru

Официальный сайт Академии ФСБ России

www.academy.fsb.ru

Теперь мы есть в Instagram

cryptolymp.ru

*Вход на территорию с электронной техникой – ноутбуками и планшетными компьютерами – запрещен.

Межрегиональная олимпиада школьников имени И.Я. Верченко

Подробное руководство

Приглашаем учащихся 8-11 классов принять участие в олимпиадах школьников

Межрегиональная олимпиада школьников имени И.Я. Верченко

по математике и криптографии

отборочный этап
с 01 ноября 2020г. по 22 ноября 2020 г.

заключительный этап
29 ноября 2020 г.

Межрегиональная олимпиада школьников имени И.Я. Верченко

по информатике и компьютерной безопасности

отборочный этап
с 1 декабря 2020 г. по 24 января 2021 г.

заключительный этап
31 января 2021 г.

Отборочный этап

Отборочные этапы олимпиад проходят в дистанционном формате на сайте www.v-olymp.ru.

На решение задач отборочного этапа нужно выделить 4 часа в пределах сроков проведения. Перед началом решения задач рекомендуем ознакомиться с предлагаемым разбором решений олимпиады прошлого года.

Результаты отборочного этапа размещаются в личных кабинетах участников после окончания этапа.

Заключительный этап

В случае успешного прохождения отборочного этапа в личном кабинете участника на сайте www.v-olymp.ru Вы сможете зарегистрироваться на заключительный этап и распечатать комплект участника (анкета, листы для выполнения письменной работы).

Заключительный этап проводится в очном формате.

С собой необходимо взять: паспорт РФ, письменные принадлежности.

Результаты и графические образы проверенных работ будут доступны участникам в их личных кабинетах на сайте www.v-olymp.ru.

Льготы при поступлении в вузы

Олимпиада школьников по иностранным языкам и все олимпиады технического профиля традиционно включаются в Перечень олимпиад школьников на учебный год, утверждаемый Министерством образования и науки России.

Включение в Перечень позволяет вузам предоставлять особые права при поступлении призерам и победителям указанных олимпиад:

- принять победителя (призера) олимпиады без вступительных испытаний.
- приравнять победителя (призера) олимпиады к школьникам, набравшим максимальное количество баллов за ЕГЭ или дополнительное профильное испытание.

Вышеуказанные особые права предусмотрены правилами приема в Академию ФСБ России в текущем учебном году.

Как подготовиться

Всем участникам предоставляется возможность прохождения ознакомительного этапа.

Традиционно задания этапа составляются на основании отборочного этапа прошлого года. Ознакомительный этап позволяет школьникам освоить интерфейс системы проведения отборочного этапа, а также попробовать себя в решении задач олимпиадной тематики.

Ознакомительный этап носит факультативный характер, и факт участия/неучастия в нем, а также полученные в его ходе баллы, никак не влияют на результаты последующих этапов.

На сайте www.v-olymp.ru работают дистанционные курсы подготовки к олимпиадам.

Школьникам предоставляется возможность ознакомления с основными идеями решений задач прошлых олимпиад, что позволяет им почувствовать специфику олимпиады и повысить свой образовательный уровень.

СБОРНИК ЗАДАЧ МЕЖРЕГИОНАЛЬНОЙ ОЛИМПИАДЫ ШКОЛЬНИКОВ ИМЕНИ И.Я. ВЕРЧЕНКО ПО ИНФОРМАТИКЕ И КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ ЗА 2019-2020 ГОДЫ С РЕШЕНИЯМИ

Москва 2020

Сборник задач межрегиональной олимпиады школьников имени И.Я. Верченко по информатике и компьютерной безопасности за 2019-2020 годы с решениями. – М., 2020. – 44 с.

Традиционно, начиная с 2006 года, Академией ФСБ России и Федеральным учебно-методическим объединением в системе высшего образования по укрупнённой группе специальностей и направлений подготовки 10.00.00 «Информационная безопасность» ежегодно проводится Межрегиональная олимпиада школьников имени И.Я. Верченко по информатике и компьютерной безопасности.

С 2016/17 учебного года Олимпиаде присваивается 2-й уровень по профилю «Компьютерная безопасность» и специальностям укрупнённой группы «Информационная безопасность».

Сборник содержит примеры задач, которые были предложены учащимся 8-11 классов на олимпиадах, проводимых в 2019/20 учебном году. При разборе решений читатель получает возможность познакомиться с некоторыми элементами методов защиты информации в компьютерных системах.

Сборник задач может быть использован не только при подготовке к олимпиадам, но и при изучении дисциплины «Информатика» и других дисциплин по специальностям укрупнённой группы «Информационная безопасность».

СОДЕРЖАНИЕ

8-10 КЛАССЫ

Задача 1. Парольная комбинация

Для входа в систему используется пароль, состоящий из трёх двузначных чисел, расположенных следующим образом:

```
xx-xx-xx
```

Известно, что пароль состоит из *3-х неповторяющихся простых чисел*. При этом, *последняя цифра первого числа равна первой цифре второго числа*, а *последняя цифра второго числа равна первой цифре третьего числа*.

Пример:

```
x1-17-7y
```

Задержка между попытками входа в систему равна 1 секунде. За какое *минимальное* время (в секундах) можно *гарантированно* получить пароль, если на ввод пароля время не тратится, и количество попыток ввода пароля не ограничено?

Решение

Для начала необходимо найти все простые числа в диапазоне от 11 до 99 (двузначные числа). *Простым* является такое число, которое делится только на себя и на 1. Сделать это можно перебором, проверяя, не делится ли проверяемое число на какое-либо от 2-х до самого числа (либо половины числа).

Для ускорения напишем программу, которая выводит на экран все простые числа в указанном диапазоне.

Листинг программы на языке Python.

```
# Функция, определяющая, является ли число простым
# ПАРАМЕТР:
#  n - проверяемое число
# ВОЗВРАЩАЕТ:
#  True - число n простое
#  False - число n непростое
def isprime(n):
    if n == 1:
        return True
    for d in range(2, n//2):
        if n % d == 0:
            return False
    return True

# Цикл перебора чисел от 11 до 99
# Для каждого числа вызывается функция isprime()
# Если функция вернула True - число добавляется
# в массив list_input
# Счетчик Count считает количество простых чисел
Count = 0
list_input = []
for x in range(11,100):
    if isprime(x) == True:
        list_input.append(x)
        Count += 1
print(list_input)
print("Total:", Count)
```

Листинг программы на языке C.

```
// Функция, определяющая, является ли число простым
// ПАРАМЕТР:
//  n - проверяемое число
// ВОЗВРАЩАЕТ:
//  true - число n простое
//  false - число n непростое
bool isprime(int n)
{
```

```
   if (n == 1)
      return true;
   for (int d = 2; d < n / 2; d++)
      if (n % d == 0)
         return false;
   return true;
}

int main()
{
// Цикл перебора чисел от 11 до 99
// Для каждого числа вызывается функция isprime()
// Если функция вернула true - число добавляется
// в массив
// Счетчик Count считает количество простых чисел
    int Count = 0;
    int list_input[100] = { 0 };
    // Формирование массива простых чисел
    for (int x = 11; x < 100; x++)
    {
        if (isprime(x) == true)
        {
            list_input[Count] = x;
            Count += 1;
        }
    }
    // Вывод на экран
    for (int i = 0; i < Count; i++)
    {
        printf("%d ", list_input[i]);
    }
    printf("\nTotal: %d\n", Count);
    return 0;
}
```

Результат работы программы:

```
11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73
79 83 89 97
Total: 21
```

Всего 21 простых чисел в диапазоне от 11 до 99. Теперь необходимо выделить все возможные комбинации, удовлетворяющие условию «*последняя цифра первого числа равна первой цифре второго числа*, а *последняя цифра второго числа равна первой цифре третьего числа*».

Сделать это можно перебором всех комбинаций. В цикле перебираем всевозможные числа и проверяем два условия:

- все числа разные;
- последняя цифра первого числа равна первой цифре второго числа, а последняя цифра второго числа равна первой цифре третьего числа.

Если все условия выполнены, то увеличиваем значение счетчика. В результате счетчик будет содержать количество комбинаций, удовлетворяющих заданию. Поскольку задержка между вводом паролей равна 1 секунде, а после ввода последней комбинации задержки нет, то ответ – количество найденных комбинаций минус 1.

Пример программы представлен в листингах ниже.

Листинг программы на языке Python.

```
# Функция возвращает список всех комбинаций чисел,
# удовлетворяющих заданному условию паролей
# ПАРАМЕТР:
#    list_input - список чисел, из которых будет
#                 формироваться комбинация
# ВОЗВРАЩАЕТ:
#    list_combination - список комбинаций
#             из 3-х чисел, удовлетворяющих условию
def combination(list_input):
  list_combination = []
  for a in list_input:
    for b in list_input:
      for c in list_input:
        if a != b and b != c and a != c:
```

```
            if str(a)[-1] == str(b)[0] and
               str(b)[-1] == str(c)[0]:
              list_combination.append(str(a) +
                      "-" + str(b) + "-" + str(c))
    return list_combination

# Заполнение списка всеми простыми числами
# в заданном диапазоне
list_input = []
for x in range(11,100):
    if isprime(x) == True:
        list_input.append(x)
# Список, содержащий все комбинации,
# удовлетворяющие заданному условию
result_list = combination(list_input)
# Вывод всех комбинаций
for i in result_list:
    print(i)
# Вывод количества комбинаций
print("Total:", len(result_list))
```

Листинг программы на языке С.

```
// Функция выводит на экран все комбинации,
// удовлетворяющие условию
// ПАРАМЕТРЫ:
//    list_input - массив чисел, из которых будут
//                 строится комбинации
//    count - количество чисел в массиве list_input
// ВОЗВРАЩАЕТ:
//    выводит на экран комбинации, удовлетворяющие
//    условию
//  выводит на экран количество комбинаций
void combination(int list_input[], int count)
{
   int last_digit1;
   int first_digit2;
   int last_digit2;
   int first_digit3;
   int total = 0;
```

```
   for (int i = 0; i < count; i++)
      for (int j = 0; j < count; j++)
         for (int k = 0; k < count; k++)
         {
            if ( list_input[i] != list_input[j] &&
                 list_input[j] != list_input[k] &&
                 list_input[i] != list_input[k] )
            {
               // последняя цифра числа
               last_digit1 = list_input[i] % 10;
               // первая цифра числа
               first_digit2 = list_input[j] / 10;
               last_digit2 = list_input[j] % 10;
               first_digit3 = list_input[k] / 10;
               if ( last_digit1 == first_digit2 &&
                    last_digit2 == first_digit3 )
               {
                  printf("%d-%d-%d\n",
                        list_input[i], list_input[j],
                        list_input[k]);
                  total++;
               }
            }
         }
   // Вывод общего количества комбинаций
   printf("Total: %d\n", total);
}
```

В результате выполнения программы получим следующие данные:

```
11-13-31, 11-13-37, 11-17-71, 11-17-73, 11-17-79,
11-19-97, 13-31-11, 13-31-17, 13-31-19, 13-37-71,
13-37-73, 13-37-79, 17-71-11, 17-71-13, 17-71-19,
17-73-31, 17-73-37, 17-79-97, 19-97-71, 19-97-73,
19-97-79, 23-31-11, 23-31-13, 23-31-17, 23-31-19,
23-37-71, 23-37-73, 23-37-79, 29-97-71, 29-97-73,
29-97-79, 31-11-13, 31-11-17, 31-11-19, 31-13-37,
31-17-71, 31-17-73, 31-17-79, 31-19-97, 37-71-11,
37-71-13, 37-71-17, 37-71-19, 37-73-31, 37-79-97,
```

```
41-11-13, 41-11-17, 41-11-19, 41-13-31, 41-13-37,
41-17-71, 41-17-73, 41-17-79, 41-19-97, 43-31-11,
43-31-13, 43-31-17, 43-31-19, 43-37-71, 43-37-73,
43-37-79, 47-71-11, 47-71-13, 47-71-17, 47-71-19,
47-73-31, 47-73-37, 47-79-97, 53-31-11, 53-31-13,
53-31-17, 53-31-19, 53-37-71, 53-37-73, 53-37-79,
59-97-71, 59-97-73, 59-97-79, 61-11-13, 61-11-17,
61-11-19, 61-13-31, 61-13-37, 61-17-71, 61-17-73,
61-17-79, 61-19-97, 67-71-11, 67-71-13, 67-71-17,
67-71-19, 67-73-31, 67-73-37, 67-79-97, 71-11-13,
71-11-17, 71-11-19, 71-13-31, 71-13-37, 71-17-73,
71-17-79, 71-19-97, 73-31-11, 73-31-13, 73-31-17,
73-31-19, 73-37-71, 73-37-79, 79-97-71, 79-97-73,
83-31-11, 83-31-13, 83-31-17, 83-31-19, 83-37-71,
83-37-73, 83-37-79, 89-97-71, 89-97-73, 89-97-79,
97-71-11, 97-71-13, 97-71-17, 97-71-19, 97-73-31,
97-73-37
Total: 126
```

Всего таких комбинаций будет 126, следовательно, максимальное затраченное время будет равно

$$(126 - 1) * 1 \text{ сек} = 125 \text{ сек.}$$

Ответ: минимальное время, за которое можно гарантированно получить пароль, равно **125 секунд.**

Задача 2. Сетевой трафик

Был получен фрагмент сетевого трафика пользователя при взаимодействии с игровым сервером. Известно, что сервер работает по протоколу UDP и его порт назначения равен 8229. Структура UDP-дейтаграммы представлена ниже:

2 байта	2 байта	2 байта	2 байта	…
UDP-порт отправителя	UDP-порт получателя	Длина UDP-дейтаграммы	Контрольная сумма	Данные

Длина UDP-дейтаграммы включает в себя размер заголовка и размер данных в байтах.

Дамп трафика:

```
0B D7 20 25 00 16 1D DC 47 45 54 20 43 4F 4D 4D 41
4E 44 3A 20 25 20 25 0B D7 00 12 69 AF 53 45 54 20
43 4F 4F 52 44 3A 20 25 0B D7 00 12 25 C0 20 25 28
33 34 2C 35 34 29 00 0B D7 20 25 00 14 2C 8F 43 4F
4D 4D 41 4E 44 20 2D 20 4F 4B
```

Определите, какие данные *сервер отправил клиенту*.

Решение

Из условия задачи известно, что взаимодействие происходит между клиентом и сервером. Для адресации используется поле UDP-порт отправителя и UDP-порт получателя (на каждый из них выделяется по 2 байта). При этом, если порт сервера равен 8229, то в дейтаграммах, отправленных клиентом, поле «UDP-порт получателя» будет равно 8229. В дейтаграммах, отправленных сервером, поле «UDP-порт отправителя» будет равно 8229.

Число $8229_{10} = 2025_{16}$.

Анализируя трафик, можно заметить, что 3-й и 4-й байты равны «2025» – это поле «UDP-порт получателя». Значит, первая дейтаграмма отправлена клиентом серверу. Необходимо найти размер этой дейтаграммы (5-6 байты):

$$0016_{16} = 22_{10}.$$

Следующая дейтаграмма начинается с 23-го байта:

```
2025 0BD7 0012 69AF …
```

Как видно, 1-2 байты дейтаграммы равны 20 25 – это поле «UDP-порт отправителя». Это означает, что данная дейтаграмма была отправлена сервером клиенту. Размер данной дейтаграммы: $0012_{16} = 18_{10}$.

Вся дейтаграмма выглядит следующим образом:

```
2025 0BD7 0012 69AF 53 45 54 20 43 4F 4F 52 44 3A
```

Поле данных (10 байт): `53 45 54 20 43 4F 4F 52 44 3A`.

В соответствии с ASCII-таблицей, каждому байту соответствует символ. Получаем следующую строку:

SET COORD:

Анализируя дальнейший дамп трафика, можно заметить, что следующая дейтаграмма также начинается с байтов 2025, это означает, что следующая дейтаграмма также отправлена сервером клиенту:

```
2025 0BD7 0012 25C0 20 25 28 33 34 2C 35 34 29 00
```

Размер дейтаграммы равен $0012_{16} = 18_{10}$.

Поле данных (10 байт): `20 25 28 33 34 2C 35 34 29 00`

В соответствии с ASCII-таблицей, каждому байту соответствует символ. Получаем следующую строку:

_%(34,54)

Следующая дейтаграмма:

```
0BD7 2025 0014 2C8F 43 4F 4D 4D 41 4E 44 20 2D 20
4F 4B
```

Поле «UDP-порт назначения» равно 2025 – эта дейтаграмма отправлена клиентом серверу. Размер дейтаграммы $0014_{16} = 20_{10}$.

Больше дейтаграмм в дампе нет.

Таким образом, от сервера клиенту было отправлено две дейтаграммы со следующим содержимым:

SET COORD:_%(34,54)

Ответ: SET COORD:_%(34,54).

Задача 3. Шифрование

В системе используется следующий алгоритм шифрования текстовых сообщений: значение каждого следующего байта циклически сдвигается побитно влево *N* раз, где *N* – значение предыдущего зашифрованного байта. Первый байт сообщения не шифруется.

Расшифруйте предоставленный зашифрованный фрагмент текста:

```
53 2B 73 23 01 C2 D5 8E 1A 80 72 95 2E 5D AC 37 3A
```

Решение

Приведенный в задании алгоритм подразумевает, что первый байт не шифруется – $53_{16} = 83_{10}$, что соответствует символу *'S'*.

Следующий символ был зашифрован путем циклического сдвига влево 83 раза. Стоит отметить, что если значение байта циклически сдвинуть влево 8 раз, получится исходное значение байта. Таким образом, сдвиг влево 83 раза равен сдвигу влево 3 раза (83 % 8 = 3, где % – остаток от деления).

Для того, чтобы получить исходное значение байта, сдвинутого влево 3 раза, необходимо его еще сдвинуть влево 8 – 3 = 5 раз (см. таблицу).

Сдвиг	Значение байта в 2-ом формате	Значение байта в 16-ом формате
0	00101011	2B
1	01010110	56
2	10101100	AC
3	01011001	59
4	10110010	B2
5	01100101	65

Сверившись с ASCII-таблицей, коду 65_{16} соответствует символ '*e*'.

Следующий байт был сдвинут влево $2B_{16} = 43_{10}$ раз, что эквивалентно сдвигу влево 3 раза (43 % 8 = 3).

Чтобы восстановить значение байта, необходимо сдвинуть его влево 5 раз.

Сдвиг	Значение байта в 2-ом формате	Значение байта в 16-ом формате
0	01110011	73
1	11100110	E6
2	11001101	CD
3	10011011	9B
4	00110111	37
5	01101110	6E

Сверившись с ASCII-таблицей, коду $6E_{16}$ соответствует символ '*n*'.

Для ускорения процесса расшифрования можно написать программу. Листинги программы представлены ниже.

Листинг программы на языке Python.

```
# Функция циклического сдвига влево
# ПАРАМЕТРЫ:
#     s - число (байт)
```

```
#    n - сколько раз сдвигать число s
# РЕЗУЛЬТАТ:
#    res - циклически сдвинутое влево число s n раз
def shift(s, n):
    res = int(s)
    for i in range(0, n):
        t = res
        # сдвиг влево на 1 бит
        res = (t << 1) & 0xFF
        # на место младшего бита записываем старший,
        # который был до сдвига
        res = res | (t >> 7)
    return res

# Функция расшифрования массива чисел
# ПАРАМЕТР:
#    cmass - массив чисел
# ВОЗВРАЩАЕТ:
#    расшифрованную строку
def decipher(cmass):
    # Результирующий текст
    text = ""
    # Копируем первый символ без изменений
    text += chr(cmass[0])
    # Цикл по оставшимся числам
    for i in range(1, len(cmass)):
        # Сдвигаем очередной символ столько раз,
        # чтобы суммарный сдвиг с учетом шифрования
        # был кратен 8
        n = 8 - (cmass[i-1] % 8)
        s = shift(cmass[i], n)
        # s преобразуем в символ
        # и добавляем в конец строки результата
        text += chr(s)
    return text

mass = [0x53, 0x2B, 0x73, 0x23, 0x01, 0xC2, 0xD5,
0x8E, 0x1A, 0x80, 0x72, 0x95, 0x2E, 0x5D, 0xAC,
0x37, 0x3A]
```

```
text = decipher(mass)
print(text)
```

Листинг программы на языке С.

```
// Функция циклического сдвига влево
// ПАРАМЕТРЫ:
//  s - число(байт)
//  n - сколько раз сдвигать число s
// РЕЗУЛЬТАТ:
//  res - циклически сдвинутое влево число s n раз
char shift(unsigned char a, int n)
{
   char res = a;
   for (int i = 0; i < n; i++)
   {
      res = a << 1;
      res |= a >> 7;
      a = res;
   }
   return res;
}

// Функция расшифрования массива чисел
// ПАРАМЕТР:
//    cmass - массив чисел
//    len - размер массива
// ВОЗВРАЩАЕТ:
//    расшифрованную строку
char* decipher(unsigned char* cmass, int len)
{
   int n;
   // выделение памяти
   char* text = new char[len+1];
   // первый элемент без изменений
   text[0] = cmass[0];
   // цикл по остальным числам из cmass
   for (int i = 1; i < len; i++)
   {
```

```
        // Сдвигаем очередной символ столько раз,
        // чтобы суммарный сдвиг с учетом шифрования
        // был кратен 8
        n = 8 - (cmass[i-1] % 8);
        text[i] = shift(cmass[i], n);
    }
    // добавляем к text нуль-символ конца строки
    text[len] = '\0';
    return text;
}
```

При подаче на вход программы последовательности из задания, получится следующий результат:

Send auth request

Данная фраза и есть ответ.

Ответ: Send auth request.

Задача 4. Стеганография

Аналитику удалось обнаружить папку с графическими изображениями, в которой скрыто осмысленное кодовое слово. Помогите определить кодовое слово, если известно, что для его сокрытия содержимое файлов не менялось.

К задаче прилагается: *папка с файлами* *(1.jpg, 2.jpg, ... , 32.jpg)* (см. рисунок).

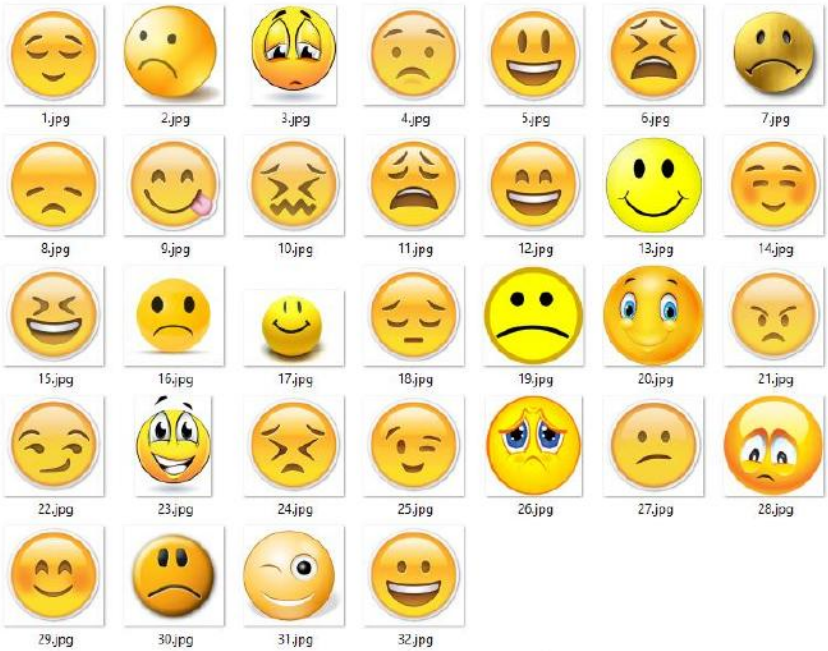


Рисунок. Прилагаемые изображения

Решение

Содержимое изображений не изменялось. Следовательно, содержимое файлов детально изучать нецелесообразно.

Все изображения представляют собой различные смайлики. Внимательно просмотрев все изображения, можно заметить, что изображенные смайлики либо грустные, либо веселые. Предположим, что каждая из картинок – бит информации, тогда можно соотнести одну категорию с единицей, а вторую – с нулем.

Всего 32 изображения – получается комбинация из 32-х нулей и единиц (32 бита).

Зная, что в байте 8 бит, а также тот факт, что каждому символу в ASCII-таблице соответствует 1 байт, можно предположить, что в картинках скрыто 4-символьное сообщение.

Предположим, что грустному смайлику соответствует «0», а веселому – «1». Получается следующая комбинация:

```
1000 1000
1001 1110
1001 0110
1000 1011
```

Переведем в 16-й формат: `88 9E 96 8B`.

Согласно ASCII-таблице, эти коды не соответствуют ни буквам, ни цифрам. Такое сообщение не несет смысл. Значит, исходное предположение было неверным.

Предположим теперь наоборот: грустному смайлику соответствует «1», а веселому – «0». Получается следующая комбинация:

```
0111 0111
0110 0001
0110 1001
0111 0100
```

Переведем в 16-й формат: `77 61 69 74`.

Согласно ASCII-таблице, эти коды соответствуют следующим символам: *wait* – что является осмысленным словом и ответом на задачу.

Ответ: wait.

Задача 5. Web-сайт

На компьютере нарушителя было найдено множество архивов, каждый из которых защищён некоторым паролем. Проведённый анализ показал, что только *три* архива содержат полезную информацию: *один* содержит «скрытое сообщение», а *два других* – фрагменты пароля к этому архиву. Остальные архивы пустые и не содержат важной информации.

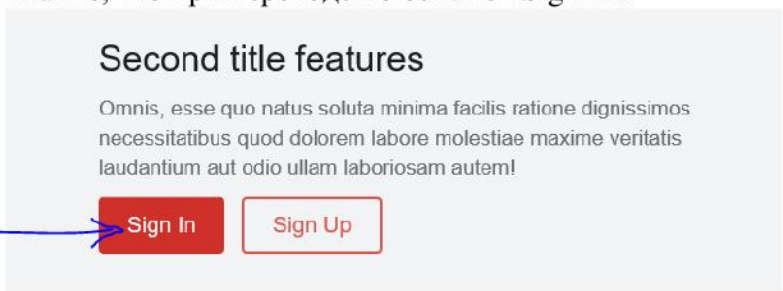
Всю необходимую информацию для доступа к архивам и секретному сообщению нарушитель спрятал на Web-странице. Определите «скрытое сообщение».

К задаче прилагается:
1) папка с содержимым web-страницы,
2) папка с архивами (архив1.rar, архив2.rar, … , архив30.rar).

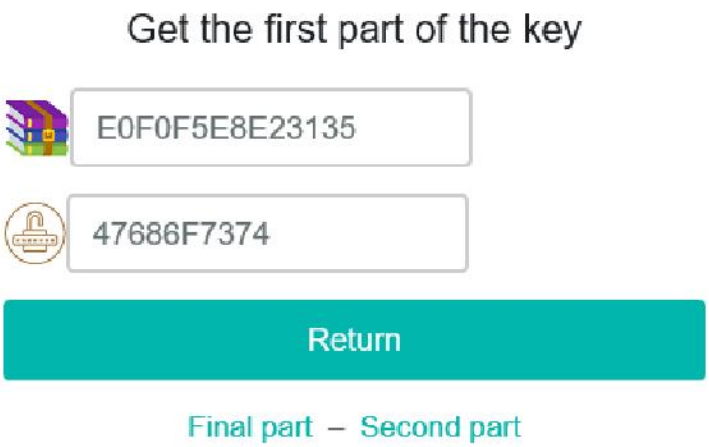
Решение

Обращаем внимание, что среди архивов ровно 3 имеют отличные от остальных дату и время модификации: *архив8.rar*, *архив15.rar*, *архив21.rar*.

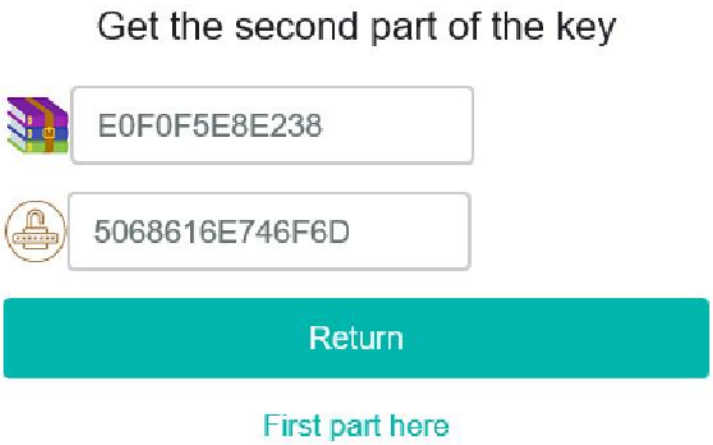
При анализе содержимого web-страницы обращаем внимание, что при переходе по ссылке «Sign-In»



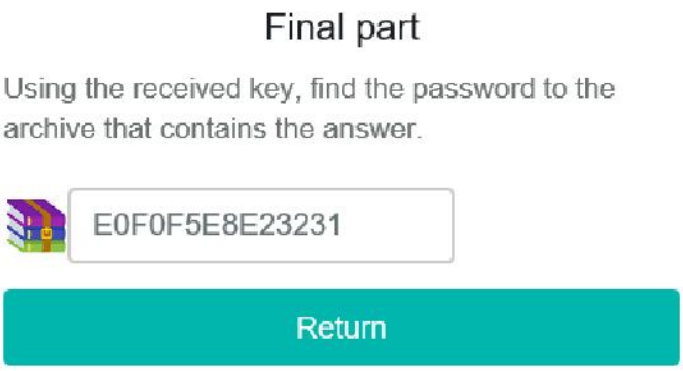
открывается форма, в который поля, обычно предназначенные для ввода логина и пароля, заполнены какой-то полезной информацией.



Перейдя по ссылкам «Second Part», получаем похожую картину:



Переход по ссылке «Final part» даёт информацию только об имени архива:



Исходя из условия задачи, можно предположить, что первые две части относятся к архивам, содержащим

фрагменты пароля к финальному архиву с искомым «скрытым сообщением».

Проанализируем полученную информацию.

На первой форме (SignIn) содержится 2 сообщения:

- `0xE0F0F5E8E23135 (E0 F0 F5 E8 E2 31 35)`, что по ASCII-таблице соответствует строке «*архив15*»,
- в поле «пароль» указан пароль `0x47686F7374 (47 68 6F 73 74)`, то соответствует строке «*Ghost*».

Можно предположить, что к файлу архив15.rar паролем является Ghost (с учетом регистра). В архиве содержится файл с изображением первой части пароля от архива со «скрытым сообщением»: *World*.

На второй форме (Second Part) так же содержатся 2 сообщения:

- `0xE0F0F5E8E238` = «*архив8*»,
- `0x5068616E746F6D` = «*Phantom*», что является паролем к файлу архив8.rar.

В архиве8 содержится файл с изображением второй части пароля от архива со «скрытым сообщением»: *Map*.

На третьей форме (Final Part) содержится только одно сообщение: `0xE0F0F5E8E23231` = «*архив21*». Можно сделать вывод, что ответ содержится в файле архив21.rar, пароль к которому «складывается из двух частей» – *WorldMap*.

В архиве содержится файл, внутри которого можно найти «скрытое сообщение» – «*Sense of security*», которое и является ответом.

Ответ: Sense of security

11 КЛАСС

Задача 1. Парольная комбинация

Для входа в систему используется пароль, состоящий из трёх шестизначных чисел, расположенных следующим образом:

`xxxxxx-xxxxxx-xxxxxx`

Известно, что пароль состоит из *3-х неповторяющихся простых чисел* в диапазоне от 100 000 до 100 300. При этом *последняя цифра первого числа равна первой цифре второго числа, а последняя цифра второго числа равна первой цифре третьего числа*.

Пример:

`xxxxx3-3xxxx7-7xxxxx`

Задержка между попытками входа в систему равна 1 секунде. За какое максимальное количество времени (в секундах) можно гарантированно получить пароль, если на ввод пароля время не тратится?

Решение

Для начала необходимо найти все простые числа в диапазоне от 100 000 до 100 300. *Простым* является такое число, которое делится только на себя и на 1. Сделать это можно перебором, проверяя, не делится ли проверяемое число на какое-либо от 2-х до самого числа (либо половины числа).

Для ускорения напишем программу, которая выводит на экран все простые числа в указанном диапазоне.

Листинг программы на языке Python.

```
# Функция, определяющая, является ли число простым
# ПАРАМЕТР:
#  n - проверяемое число
# ВОЗВРАЩАЕТ:
#  True - число n простое
#  False - число n непростое
def isprime(n):
    if n == 1:
        return True
    for d in range(2, n//2):
        if n % d == 0:
            return False
    return True

# Цикл перебора чисел от 100 000 до 100 300
# Для каждого числа вызывается функция isprime()
# Если функция вернула True - число добавляется
# в массив list_input
# Счетчик Count считает количество простых чисел
Count = 0
list_input = []
for x in range(100000,100300):
    if isprime(x) == True:
        list_input.append(x)
        Count += 1
print(list_input)
print("Total:", Count)
```

Листинг программы на языке C.

```
// Функция, определяющая, является ли число простым
// ПАРАМЕТР:
//  n - проверяемое число
// ВОЗВРАЩАЕТ:
//  true - число n простое
//  false - число n непростое
bool isprime(int n)
{
```

```
   if (n == 1)
      return true;
   for (int d = 2; d < n / 2; d++)
      if (n % d == 0)
         return false;
   return true;
}

int main()
{
// Цикл перебора чисел от 100000 до 100300
// Для каждого числа вызывается функция isprime()
// Если функция вернула true - число добавляется
// в массив
    // Счетчик Count считает количество простых чисел
    int Count = 0;
    int list_input[100] = { 0 };
    // Формирование массива простых чисел
    for (int x = 100000; x < 100300; x++)
    {
        if (isprime(x) == true)
        {
            list_input[Count] = x;
            Count += 1;
        }
    }
    // Вывод на экран
    for (int i = 0; i < Count; i++)
    {
        printf("%d ", list_input[i]);
    }
    printf("\nTotal: %d\n", Count);
    return 0;
}
```

Результат работы программы:

```
100003 100019 100043 100049 100057 100069 100103
100109 100129 100151 100153 100169 100183 100189
```

```
100193 100207 100213 100237 100267 100271 100279
100291 100297
Total: 23
```

Всего 23 простых чисел в диапазоне от 11 до 99. Теперь необходимо выделить все возможные комбинации, удовлетворяющие условию «*последняя цифра первого числа равна первой цифре второго числа, а последняя цифра второго числа равна первой цифре третьего числа*».

Сделать это можно двумя способами.

Способ 1. Перебор.

В цикле перебираем всевозможные числа и проверяем два условия:

- все числа разные;
- последняя цифра первого числа равна первой цифре второго числа, а последняя цифра второго числа равна первой цифре третьего числа.

Если все условия выполнены, то увеличиваем значение счетчика. В результате счетчик будет содержать количество комбинаций, удовлетворяющих заданию. Поскольку задержка между вводом паролей равна 1 секунде, а после ввода последней комбинации задержки нет, то ответ – количество найденных комбинаций минус 1.

Пример программы представлен в листингах ниже.

Листинг программы на языке Python.

```
# Функция возвращает список всех комбинаций чисел,
# удовлетворяющих заданному условию паролей
# ПАРАМЕТР:
#    list_input - список чисел, из которых будет
#                 формироваться комбинация
# ВОЗВРАЩАЕТ:
#    list_combination - список комбинаций
#             из 3-х чисел, удовлетворяющих условию
```

```
def combination(list_input):
  list_combination = []
  for a in list_input:
    for b in list_input:
      for c in list_input:
        if a != b and b != c and a != c:
          if str(a)[-1] == str(b)[0] and
             str(b)[-1] == str(c)[0]:
            list_combination.append(str(a) +
                  "-" + str(b) + "-" + str(c))
  return list_combination

# Заполнение списка всеми простыми числами
# в заданном диапазоне
list_input = []
for x in range(100000,100300):
    if isprime(x) == True:
        list_input.append(x)
# Список, содержащий все комбинации,
# удовлетворяющие заданному условию
result_list = combination(list_input)
# Вывод всех комбинаций
for i in result_list:
    print(i, end=", ")
# Вывод количества комбинаций
print("Total:", len(result_list))
```

Листинг программы на языке С.

```
// Функция выводит на экран все комбинации,
// удовлетворяющие условию
// ПАРАМЕТРЫ:
//    list_input - массив чисел, из которых будут
//                 строится комбинации
//    count - количество чисел в массиве list_input
// ВОЗВРАЩАЕТ:
//    выводит на экран комбинации, удовлетворяющие
//    условию
//  выводит на экран количество комбинаций
```

```
void combination(int list_input[], int count)
{
   int last_digit1;
   int first_digit2;
   int last_digit2;
   int first_digit3;
   int total = 0;
   for (int i = 0; i < count; i++)
      for (int j = 0; j < count; j++)
         for (int k = 0; k < count; k++)
         {
            if ( list_input[i] != list_input[j] &&
                 list_input[j] != list_input[k] &&
                 list_input[i] != list_input[k] )
            {
               // последняя цифра числа
               last_digit1 = list_input[i] % 10;
               // первая цифра числа
               first_digit2 = list_input[j]/100000;
               last_digit2 = list_input[j] % 10;
               first_digit3 = list_input[k]/100000;
               if ( last_digit1 == first_digit2 &&
                    last_digit2 == first_digit3 )
               {
                  printf("%d-%d-%d\n",
                        list_input[i], list_input[j],
                        list_input[k]);
                  total++;
               }
            }
         }
   // Вывод общего количества комбинаций
   printf("Total: %d\n", total);
}
```

В результате выполнения программы получим следующие данные:

```
100151-100271-100003, 100151-100271-100019,
100151-100271-100043, 100151-100271-100049,
100151-100271-100057, 100151-100271-100069,
```

100151-100271-100103, 100151-100271-100109,
100151-100271-100129, 100151-100271-100153,
100151-100271-100169, 100151-100271-100183,
100151-100271-100189, 100151-100271-100193,
100151-100271-100207, 100151-100271-100213,
100151-100271-100237, 100151-100271-100267,
100151-100271-100279, 100151-100271-100291,
100151-100271-100297, 100151-100291-100003,
100151-100291-100019, 100151-100291-100043,
100151-100291-100049, 100151-100291-100057,
100151-100291-100069, 100151-100291-100103,
100151-100291-100109, 100151-100291-100129,
100151-100291-100153, 100151-100291-100169,
100151-100291-100183, 100151-100291-100189,
100151-100291-100193, 100151-100291-100207,
100151-100291-100213, 100151-100291-100237,
100151-100291-100267, 100151-100291-100271,
100151-100291-100279, 100151-100291-100297,
100271-100151-100003, 100271-100151-100019,
100271-100151-100043, 100271-100151-100049,
100271-100151-100057, 100271-100151-100069,
100271-100151-100103, 100271-100151-100109,
100271-100151-100129, 100271-100151-100153,
100271-100151-100169, 100271-100151-100183,
100271-100151-100189, 100271-100151-100193,
100271-100151-100207, 100271-100151-100213,
100271-100151-100237, 100271-100151-100267,
100271-100151-100279, 100271-100151-100291,
100271-100151-100297, 100271-100291-100003,
100271-100291-100019, 100271-100291-100043,
100271-100291-100049, 100271-100291-100057,
100271-100291-100069, 100271-100291-100103,
100271-100291-100109, 100271-100291-100129,
100271-100291-100151, 100271-100291-100153,
100271-100291-100169, 100271-100291-100183,
100271-100291-100189, 100271-100291-100193,
100271-100291-100207, 100271-100291-100213,
100271-100291-100237, 100271-100291-100267,
100271-100291-100279, 100271-100291-100297,
100291-100151-100003, 100291-100151-100019,
100291-100151-100043, 100291-100151-100049,
100291-100151-100057, 100291-100151-100069,
100291-100151-100103, 100291-100151-100109,

```
100291-100151-100129, 100291-100151-100153,
100291-100151-100169, 100291-100151-100183,
100291-100151-100189, 100291-100151-100193,
100291-100151-100207, 100291-100151-100213,
100291-100151-100237, 100291-100151-100267,
100291-100151-100271, 100291-100151-100279,
100291-100151-100297, 100291-100271-100003,
100291-100271-100019, 100291-100271-100043,
100291-100271-100049, 100291-100271-100057,
100291-100271-100069, 100291-100271-100103,
100291-100271-100109, 100291-100271-100129,
100291-100271-100151, 100291-100271-100153,
100291-100271-100169, 100291-100271-100183,
100291-100271-100189, 100291-100271-100193,
100291-100271-100207, 100291-100271-100213,
100291-100271-100237, 100291-100271-100267,
100291-100271-100279, 100291-100271-100297
Total: 126
```

Всего таких комбинаций будет 126, следовательно, максимальное затраченное время будет равно

(126 – 1) * 1 сек = 125 сек.

Способ 2. Аналитический.

Можно заметить, что все числа из диапазона 100 000–100 300 начинаются на цифру «1». Это означает, что на место первого и второго числа подойдут только те числа, которые заканчиваются на «1». Шаблон комбинаций будет следующий:

`1xxxx1-1xxxx1-1xxxxx`

Анализируя полученные простые числа, всего 3 числа подходят на первую и вторую позицию: 100151, 100271, 100291. Учитывая невозможность повторения чисел возможны следующие варианты:

- на 1-ю позицию возможно 3 варианта,
- на вторую – 2 варианта,

– на третью – оставшиеся простые числа за вычетом использованных на первой и второй позиции (23–2=21).

Общее число комбинаций равно:

$$3 * 2 * 21 = 126.$$

Следовательно, максимальное затраченное время будет равно

$$(126 – 1) * 1 \text{ сек} = 125 \text{ сек}.$$

Ответ: минимальное время, за которое можно гарантированно получить пароль, равно **125 секунд.**

Задача 2. Сетевой трафик

Был получен фрагмент сетевого трафика пользователя при взаимодействии с игровым сервером. Известно, что сервер работает по протоколу UDP и его порт назначения равен 3365.

Структура UDP-дейтаграммы представлена ниже:

2 байта	2 байта	2 байта	2 байта	…
UDP-порт отправителя	UDP-порт получателя	Длина UDP-дейтаграммы	Контрольная сумма	Данные

Длина UDP-дейтаграммы включает в себя размер заголовка и размер данных в байтах.

Весь сетевой трафик шифруется методом *«двоичного гаммирования»*, то есть путём выполнения операции «побитового исключающего ИЛИ» между байтами UDP-дейтаграммы (включая заголовок) и байтами, полученными циклическим повторением последовательности некоторого ключа.

Дамп трафика:

```
A3 67 AA 90 A7 AD 8F 36 E0 F0 F3 95 F4 F0
E4 E7 E2 E1 87 F6 E8 F1 E2 B5 AA 90 A3 67
A7 BB BB 05 EE F1 E3 E4 E3 B5 A3 67 AA 90
A7 A7 C6 1C E4 FA E3 F0 87 98 87 FA EC B5
```

Определите, что сервер ответил на запрос пользователя, если известно, что для шифрования используется 2-байтовый ключ. В ответе укажите только содержимое поля данных UDP-дейтаграммы.

Решение

Из условия задачи известно, что взаимодействие происходит между клиентом и сервером. Можно сделать предположение, что первая дейтаграмма идет в направлении от клиента серверу. Порт клиента неизвестен, однако известен порт сервера. Следовательно, в первой дейтаграмме поле «UDP-порт получателя» (3-й и 4-й байты) будет равно $3365_{10} = 0D25_{16}$.

Весь сетевой трафик зашифрован методом «двоичного гаммирования». В основе «двоичного гаммирования» лежит операция «Исключающее ИЛИ» (XOR, ^). Таблица истинности операции «Исключающее ИЛИ»:

X	Y	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Если T – это открытый текст, K – ключ, а C – зашифрованный текст, то верно следующее:

$$T \wedge K = C$$
$$C \wedge K = T$$

При этом, $T \wedge C = K$.

Ели известны открытый текст и зашифрованный текст, то можно определить ключ шифрования.

3-й и 4-й байты первой дейтаграммы равны AA 90 – это зашифрованный текст. В этих байтах хранится значение 0D 25 – это открытый текст. Следовательно, ключ можно получить следующим образом:

$$K = AA\ 90 \wedge 0D\ 25.$$

Распишем данную операция поразрядно.

Двоичный код	Шестнадцатеричный код
10101010 10010000 00001101 00100101	AA 90 0D 25
10100111 10110101	A7 B5

Ключ шифрования – A7 B5.

Далее необходимо определить размер первой дейтаграммы. Поле «длина дейтаграммы» – это 5-й и 6-й байты – A7 AD. Однако это поле зашифровано, но нам известен ключ. Результат расшифрования поля:

Двоичный код	Шестнадцатеричный код
10100111 10101101 10100111 10110101	A7 AD A7 B5
00000000 00011000	00 18

Длина первой дейтаграммы равна $0018_{16} = 24_{10}$.

Выделим и расшифруем поле данных первой дейтаграммы. Если длина всей дейтаграммы 24 байта, а длина заголовка равна

2(порт отправителя) + 2(порт получателя) + 2(длина) + 2(контрольная сумма) = 8 байт,

то длина поля данных равна 24–8=16 байт.

Зашифрованные данные:

```
E0 F0 F3 95 F4 F0 E4 E7 E2 E1 87 F6 E8 F1 E2 B5
```

Используя ключ, расшифруем эти байты:

```
E0 F0 F3 95 F4 F0 E4 E7 E2 E1 87 F6 E8 F1 E2 B5
A7 B5 A7 B5 A7 B5 A7 B5 A7 B5 A7 B5 A7 B5 A7 B5
47 45 54 20 53 45 43 52 45 54 20 43 4F 44 45 00
```

Каждый полученный байт представляет собой код символа в ASCII-таблице. В результате получаем текстовое сообщение: «*GET SECRET CODE*».

Следующая дейтаграмма начинается с 25-го байта:

```
AA90 A367 A7BB BB05 EE F1 ...
```

По логике взаимодействия вторая дейтаграмма – это ответ сервера. Значит, в поле «UDP-порт отправителя» должно стоять значение 0D25. В зашифрованном трафике в этом поле стоит значение AA90.

Используя вычисленный ранее ключ, можно расшифровать первые 2 байта дейтаграммы:

AA90 ^ A7B5 = 0D25

Предположение подтвердилось – данная дейтаграмма отправлена от сервера клиенту. Осталось определить длину дейтаграммы (5-й и 6-й байты).

В зашифрованном виде поля длины дейтаграммы равно A7BB. Используя ключ A7B5 расшифруем это поле.

Двоичный код	Шестнадцатеричный код
10100111 10111011 10100111 10110101	A7 BB A7 B5
00000000 00001110	00 0E

Длина второй дейтаграммы равна $000E_{16} = 14_{10}$.

Выделим и расшифруем поле данных второй дейтаграммы. Если длина всей дейтаграммы 14 байт, а длина

заголовка равна 8 байт, то длина поля данных равна 14-8=6 байт.

Зашифрованные данные: `EE F1 E3 E4 E3 B5`.

Используя ключ, расшифруем эти байты:

```
EE F1 E3 E4 E3 B5
A7 B5 A7 B5 A7 B5
-----------------
49 44 44 51 44 00
```

Каждый полученный байт представляет собой код символа в ASCII-таблице. В результате получаем текстовое сообщение: «*IDDQD*».

Рассмотрим третью дейтаграмму:

```
A367 AA90 A7A7 C61C E4 FA E3 F0 87 98 87 FA EC B5
```

Анализируя поля «UDP-порт источника» и «UDP-порт назначения», можно сделать вывод, что данная дейтаграмма отправлена от клиента серверу (порт назначения равен AA90, что при расшифровании даст 0D25 – порт сервера).

Определим размер третьей дейтаграммы. Поле «длина дейтаграммы» – это 5-й и 6-й байты – A7 A7. Используя известный ключ, расшифруем это поле.

Двоичный код	Шестнадцатеричный код
10100111 10100111 10100111 10110101	A7 A7 A7 B5
00000000 00010010	00 12

Длина первой дейтаграммы равна $0012_{16} = 18_{10}$.

Выделим и расшифруем поле данных третьей дейтаграммы. Если длина всей дейтаграммы 18 байт, а длина заголовка 8 байт, то длина поля данных равна 28–8=10 байт.

Зашифрованные данные:

```
E4 FA E3 F0 87 98 87 FA EC B5
```

Используя ключ, расшифруем эти байты:

```
E4 FA E3 F0 87 98 87 FA EC B5
A7 B5 A7 B5 A7 B5 A7 B5 A7 B5
-----------------------------
43 4F 44 45 20 2D 20 4F 4B 00
```

Каждый полученный байт представляет собой код символа в ASCII-таблице. В результате получаем текстовое сообщение: «*CODE – OK*».

Ответ: сервер отправил клиенту текстовое сообщение **IDDQD**. Ключ шифрования **`A7B5`**.

Задача 3. Вирус

Имеется система, представляющая собой файл-серверную архитектуру, состоящую из 1 файл-сервера и 6 ПК. На файл-сервере хранится 30 файлов-приложений (*file1.exe, file2.exe, ..., fileN.exe*). Известно, что один из файлов заражён вредоносным кодом, который после попадания на клиентское устройство выводит его из строя через 1 час. Каждый ПК может копировать с файл-сервера любое количество файлов.

За какое *минимальное количество часов* можно точно определить зараженный файл. Ответ обоснуйте.

Решение

Для определения вредоносного файла необходимо каждый из них пронумеровать следующим образом.

1. Номер каждого файла представим в виде двоичного 6-разрядного разложения:
   file1.exe – 000001 (двоичное представление числа 1)
   file2.exe – 000010 (двоичное представление числа 2)
   file3.exe – 000011 (двоичное представление числа 3)
   …
   file30.exe – 11110 (двоичное представление числа 30)

Такое разложение позволяет каждому файлу предоставить уникальную последовательность нулей и единиц.

2. Всего 6 ПК. Распределим их номера: 0,1,2,3,4,5.
3. Каждому ПК соответствует разряд в последовательности представления файла:
   ПК0 – младший разряд (0-й),
   ПК1 – второй справа разряд (1-й),
   …
   ПК5 – старший разряд (5-й)
4. На каждый ПК копируем только те файлы, в соответствующем разряде которого стоит «1»:
   на ПК0 – file1.exe, file3.exe, file5.exe, … , file29.exe
   на ПК1 – file2.exe, file3.exe, file6.exe, file7.exe, …
   на ПК2 – file4.exe, file5.exe, file6.exe, file7.exe, …
   …
   на ПК5 – file16.exe, file17.exe, file18.exe, … , file30.exe.
5. Ровно через 1 час после копирования какие-то ПК выйдут из строя. Их номера определят номер зараженного файла следующим образом: если ПКi-й не поврежден, то на i-м разряде номера файла ставим «0». Если ПКi-й поврежден, то на i-м разряде номера файла ставим «1».
   В результате получится последовательность, переведя которую в десятичный формат можно однозначно определить зараженный файл.

Ответ: 1 час.

Задача 4. Стеганография

На web-странице содержатся ссылки на скачивание файлов-изображений с указанием контрольных сумм их содержимого. Известно, что в одном из них спрятано секретное текстовое сообщение. Определите это изображение и скрытое в нем секретное текстовое сообщение.

Файлы и контрольные суммы их содержимого (MD5):

```
d812f179df6b60943dbfd69c4e613aaf  Chrysanthemum.jpg
4e427c78ecb620aecce46bb006d247e5  Desert.jpg
6dbafbc2e49df3c3cfe7515d5c6cac72  Hydrangeas.jpg
f037c82a48be22696142c59b4eea3298  Jellyfish.jpg
e3614da88d1511dfb8a05dd3ec24999d  Koala.jpg
46b851500907f4ccdfa75c0a29dd8dcf  Lighthouse.jpg
2022b59db2185282fd753f6320e782c4  Penguins.jpg
902b6ea2111efb87b19056655165c72d  Tulips.jpg
```

К задаче прилагается:

1) 8 файлов-изображений (.jpg),*

2) программа подсчета контрольной суммы по алгоритму MD5 (fciv.exe),

3) файл с описанием формата RAR-архива.

Решение

1. С помощью программы fciv.exe определим MD5-суммы для файлов и сравним их с данными в задаче.
2. Можно заметить, что для файла *Jellyfish.jpg* контрольная сумма не совпадает.
3. Предполагаем, зачем дано описание формата RAR-файла. Возможно, это намек на то, что в файле с изображением содержится скрытый RAR-архив.

4. Из описания RAR-архива понятно, что любой архив начинается с метки 0x5052. Используя HEX-редактор анализируем побайтовое содержимое файла *Jellyfish.jpg* и осуществляем поиск метки 0x5052. Метка найдена.
5. Копируем содержимое файла, начиная с метки, в новый файл, который назовем *архив.rar*
   ИЛИ
   можно переименовать файл Jellyfish.jpg в *архив.rar*.
6. В архиве содержится файл *Сообщение.docx*, открыв который, читаем сообщение – *«Система обнаружения вторжений (IDS)»*.

Ответ: Система обнаружения вторжений (IDS) (файл – Jellyfish.jpg).

Задача 5. Разработка процессора

При разработке процессора инженеры решили реализовать поддержку виртуальной памяти, когда программа оперирует виртуальным адресом, а этот адрес пересчитывается процессором в физический адрес ячейки – байта в оперативной памяти. Инженеры решили использовать структуру виртуального адреса, состоящего из трёх частей:

Индекс-1	Индекс-2	Индекс-3

Вся память делится на непересекающиеся блоки одинаковой длины – страницы.

Индекс-1 содержит номер таблицы, в которой содержится информация о страницах.

Индекс-2 содержит номер страницы из таблицы с номером Индекс-1.

Индекс-3 определяет номер байта на странице с номером Индекс-2 из таблицы с номером Индекс-1 (см. рисунок).

Каждая таблица занимает ровно 1 страницу, размер таблиц фиксирован. Каждая запись в таблице имеет фиксированный размер и состоит из следующих полей:

– физический адрес страницы в оперативной памяти;
– заполнитель до кратности байту.

Общий диапазон адресов физической памяти, который должен адресоваться процессором – 1 Гб, размер страницы – 1 Кб.

Укажите *минимально необходимые* размерности полей Индекс-1, Индекс-2, Индекс-3, а также общую размерность виртуального адреса для возможности адресации объёма оперативной памяти в 1 Гб?

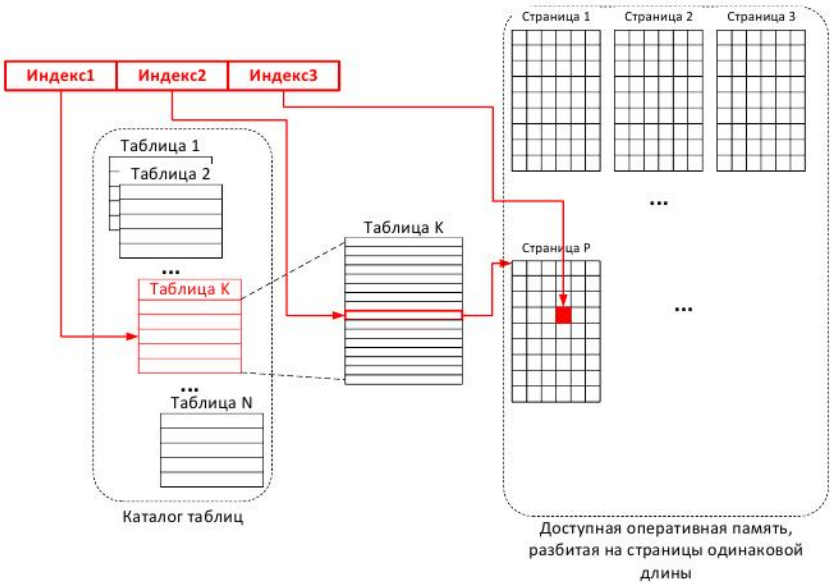


Рисунок. Организация виртуальной памяти

Решение

Вся память разбита на страницы. Страница занимает 1 Кб = 1024 байта. Значит, для адресации байта внутри страницы требуется 10 бит (2^{10} = 1024).

Индекс-3 = 10 бит.

В таблице одна запись занимает физический адрес и заполнитель. Для адресации 1 Гб адресного пространства (1 073 741 824 байт) нужен диапазон значений физического адреса [0:2^{30}-1], т.е. 30 двоичных разрядов при линейном строении адреса. Соответственно, одна запись в таблице занимает 30(адрес) + 2(заполнитель) = 32 двоичных разряда (4 байта). Размер таблицы – 1 Кб. Всего в таблице может быть 1024 / 4 = 256 записей. Для адресации 256 записей необходимо 8 двоичных разрядов (2^8 = 256).

Индекс-2 = 8 бит.

В 1 Гб помещается 1 073 741 824 / 1024 = 1 048 576 страниц размером 1 Кб. Учитывая, что в одной таблице 256 записей, всего таблиц должно быть 1 048 576 / 256 = 4096. Для адресации 4096 таблиц понадобится 12 двоичных разрядов ($2^{12} = 4096$).

Индекс-1 = 12 бит.

Итого, общая размерность виртуального адреса будет
Индекс-1 + Индекс-2 + Индекс-3 = 12 + 8 + 10 = 30 бит.

Ответ: **Индекс-1 – 12 бит,**
Индекс-2 – 8 бит,
Индекс-3 – 10 бит.
Всего – 30 бит.

СБОРНИК ЗАДАЧ МЕЖРЕГИОНАЛЬНОЙ ОЛИМПИАДЫ ШКОЛЬНИКОВ ИМЕНИ И.Я. ВЕРЧЕНКО ПО ИНФОРМАТИКЕ И КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ ЗА 2019-2020 ГОДЫ С РЕШЕНИЯМИ

XXIX
Межрегиональная олимпиада школьников им. И. Я. Верченко по математике и криптографии

Москва 2020

Всего пронумеровано 32 стр.
Подписано к печати __.01.20
Авт. л. 1,17 Усл. печ. л. 1,75
Заказ № _____.
Тираж 400 экз.

---

www.cryptolymp.ru

Приветствие главного учёного секретаря Академии криптографии Российской Федерации Владимира Николаевича Сачкова участникам XXIX Межрегиональной олимпиады школьников имени И.Я. Верченко по математике и криптографии

Дорогие друзья!

Приветствую участников XXIX Межрегиональной олимпиады школьников имени И.Я. Верченко по математике и криптографии!

Математика на протяжении всей истории человечества привлекает к себе лучшие умы и в значительной степени влияет на развитие наук о природе и технических наук. Криптография, возникшая в глубокой древности вместе с письменностью как искусство защиты сообщений от непосвящённых, прошла большой путь развития и сегодня представляет собой обширную научно-техническую область, которая опирается на достижения математики, физики, информатики, теории связи и многих других наук.

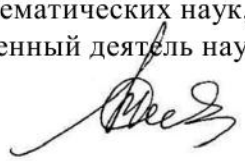
Отечественная криптография развивается вместе с нашей страной и является существенной частью системы обеспечения национальной безопасности, защиты информационных ресурсов каждого гражданина, общества и государства. В 2019 году мы отметили 70-летие важных государственных решений, их выполнение обеспечило настоящую научно-техническую революцию в

отечественной криптографии, по-прежнему обеспечивающей в этой области стратегический паритет с другими державами. В 1949 году были заложены основы и современной системы подготовки специалистов по криптографии, образованы закрытое отделение механико-математического факультета Московского государственного университета им. М.В. Ломоносова и Высшая школа криптографов – родоначальники Института криптографии, связи и информатики.

Для решения актуальных задач современной криптографии нужны в первую очередь специалисты с высоким уровнем математического образования. Олимпиады школьников играют важную роль в поиске и привлечении талантливых молодых людей, которые смогут проявить свои лучшие качества в трудном, но удивительно красивом мире математики и криптографии.

Вам, юным участникам и будущим победителям и призёрам Межрегиональной олимпиады, принадлежит будущее. Желаю всем участникам олимпиады удачи и творческих успехов!

Главный учёный секретарь Академии криптографии Российской Федерации,
доктор физико-математических наук,
профессор, заслуженный деятель науки РФ



В.Н. Сачков

Информация о проведении олимпиады

XXIX Межрегиональная олимпиада школьников им. И.Я. Верченко по математике и криптографии проводилась в два тура.

Первый тур проводился в дистанционной форме на интернет-сайте **www.cryptolymp.ru**.

Второй тур проводился в очной форме на базе Академии ФСБ России и во многих городах и вузах России: Архангельск – САФУ, Астрахань – АГУ, Астрахань – АГТУ, Барнаул – АлтГТУ, Белгород – БГТУ им. В.Г. Шухова, Владивосток – ДВФУ, Владимир – ВлГУ, Волгоград – ВолГУ, Ижевск – УдГУ, Иркутск – ИГУ, Йошкар-Ола – ПГТУ, Казань – КНИТУ-КАИ, Калининград – БФУ им. И. Канта, Кострома – КГУ, Краснодар – КубГТУ, Красноярск – СибГУ им. М.Ф. Решетнева, Курск – ЮЗГУ, Липецк – ЛГПУ имени П.П. Семенова-Тян-Шанского, Москва – Академия ФСБ России, Нефтекамск – НФБашГУ, Нижний Новгород – ННГУ им. Н.И. Лобачевского, Новосибирск – НГУЭУ, НГТУ, Озерск – ОТИ НИЯУ МИФИ, Омск – ОмГУ, Оренбург – ОГУ, Пермь – ПНИПУ, Пятигорск – ПГУ, Ростов-на-Дону – ДГТУ, Самара – Самарский университет, Санкт-Петербург – СПб НИУ ИТМО, СПбПУ, Саратов – СГТУ им. Гагарина Ю.А., Севастополь – СевГУ, Ставрополь – СКФУ, Сыктывкар – СГУ имени Питирима Сорокина, Таганрог – ЮФУ, Тамбов – ТГТУ, Томск – ТУСУР, Тюмень – ТюмГУ, Хабаровск – ДВГУПС, Челябинск – ЧелГУ, Череповец – ЧГУ, Ярославль – ЯрГУ.

Межрегиональная олимпиада школьников им. И. Я. Верченко по математике и криптографии включена в Перечень олимпиад школьников на 2019/2020 учебный год (2 уровень), что дает право предоставлять

льготы победителям и призерам при поступлении в государственные и муниципальные учреждения высшего образования (Приказ Минобрнауки России от 04.04.2014 № 267). Решения о льготах принимаются вузами самостоятельно и должны быть объявлены к 1 июня 2020 года.

УСЛОВИЯ И РЕШЕНИЯ ЗАДАЧ

Задача 1 (8-9, 10 классы)

На билетах в кинотеатры Криптоландии проставляется шестизначный номер от (0,0,0,0,0,0) до (8,8,8,8,8,8). При этом используются только цифры 0,1,2,3,4,5,6,7,8. Билет считается «счастливым», если остатки от деления на 9 суммы первых трех цифр и суммы последних трех цифр отличаются на фиксированное число $k = 2$. Например, билеты с номерами 123026 и 123661 – счастливые, а с номерами 123000 и 876111 – нет. Найдите число счастливых билетов.

Решение

Количество трёхзначных чисел $x_1x_2x_3$, у которых остаток от деления на 9 суммы цифр равен фиксированному значению $t \in \{0,1,\dots,8\}$, равно $9^2 = 81$, поскольку любые две цифры однозначно определяют третью из соотношения $r_9(x_1 + x_2 + x_3) = t$. Приведём возможные варианты для значений остатков для первой и последней тройки цифр:

$$(0,2), (1,3), \dots, (6,8),$$
$$(2,0), (3,1), \dots, (8,6)$$

их число равно $2 \times 7 = 14$, и тогда общее число счастливых билетов равно $2 \times 7 \times 9^2 \times 9^2 = 2 \times 7 \times 9^4 = 91854$.

Ответ: 91854.

Комментарий

При передаче информации по каналам связи важными аспектами являются обеспечение ее *конфиденциальности* и *целостности*. Конфиденциальность (т.е. секретность) достигается криптографическими методами, путем шифрования информации. Целостность информации означает отсутствие изменений в передаваемой/хранимой информации по сравнению с ее первоначальной записью. Говоря о целостности сообщения, по-другому говорят о его *аутентичности*.

Существует множество методов проверки целостности передаваемого сообщения. В частности, для этого широко используется теория кодирования информации. С точки зрения этой теории множество всех наборов цифр (x_1, x_2, x_3), для которых выполнено соотношение $r_m(x_1 + x_2 + x_3) = t$, является кодом с расстоянием два по Хэммингу. Этот код способен обнаруживать одну ошибку, которая может произойти при передаче набора (x_1, x_2, x_3) по каналу связи. Например, если ошибка произошла в первом символе, и было принято сообщение (x'_1, x_2, x_3), в котором $x'_1 \neq x_1$, то $r_m(x'_1 + x_2 + x_3) \neq t$ и этот факт будет обнаружен. Задача нацелена на нахождение мощности такого кода, т.е. числа всех комбинаций цифр (x_1, x_2, x_3), удовлетворяющих условию $r_m(x_1 + x_2 + x_3) = t$. Эта мощность равна m^2, поскольку при задании любых двух цифр в комбинации третья цифра определяется однозначно.

Задача 2 (8-11 классы)

Известно, что p, p_1, p_2, p_3 – различные простые числа, и $p^3 - 2p^2 - 16p = p_1 \cdot p_2 \cdot p_3 - 32$. Найдите все такие числа p, p_1, p_2, p_3. Ответ обоснуйте.

Решение

Пусть $p_1 < p_2 < p_3$. По условию $p^3 - 2p^2 - 16p + 32 = p_1 \cdot p_2 \cdot p_3$. Разложим левую часть на множители:

$$(p-2)(p-4)(p+4) = p_1 \cdot p_2 \cdot p_3. \qquad (1)$$

Непосредственной проверкой убеждаемся, что $p \neq 2,3,5$. Значит $p > 5$. Следовательно, числа в левой части (1) различны и отличны от 1. Поэтому $p - 4 = p_1$, $p - 2 = p_2$, $p + 4 = p_3$. Поскольку p на 3 не делится, возможны случаи:

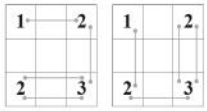
- число p при делении на 3 дает остаток 1. Тогда на 3 делится число $p - 4$. Такое возможно только, когда $p - 4 = 3$, так как число $p - 4$ простое. Отсюда $p = 7, p_1 = 3, p_2 = 5, p_3 = 11$.
- число p при делении на 3 дает остаток 2. Тогда на 3 делится $p + 4$. Значит $p + 4 = 3$, что невозможно.

Ответ: $p = 7, p_1 = 3, p_2 = 5, p_3 = 11$ (при условии $p_1 < p_2 < p_3$).

Задача 3 (8-9, 10 классы)

Сообщение передается в виде таблицы 7×7 клеток. В каждой клетке записана либо буква, либо цифра. Чтобы прочитать сообщение, необходимо зачеркнуть отрезками лишние символы. Отрезки проводят по следующим правилам (см. примеры): 1) концы отрезков лежат только в клетках с цифрами, причем цифра показывает сколько

концов в этой клетке лежит, 2) отрезки могут проходить только горизонтально или вертикально, 3) две цифры могут быть соединены не более, чем двумя отрезками. Прочитайте сообщение, которое получается выписыванием каждой третьей незачеркнутой буквы.



3	с	з	**4**	е	м	**3**
ю	с	е	р	д	е	у
ш	в	в	н	**2**	ь	**5**
о	г	д	р	б	о	ф
а	а	о	к	д	х	л
я	ж	н	т	ц	и	у
1	я	к	**2**	т	е	**2**

Решение

3	=	=	**4**	—	—	**3**
‖	**с**	е	‖	д	**е**	‖
‖	в	в	‖	**2**	=	**5**
‖	**г**	д	‖	б	**о**	‖
‖	а	о	‖	**д**	х	‖
‖	ж	**н**	‖	ц	и	‖
1	**я**	к	**2**	—	—	**2**

Для решения задачи следует для каждого числа рассматривать количество соседей – чисел, с которыми оно может соединяться отрезками.

Если число равно удвоенному количеству своих соседей, то с каждым из них оно соединяется как минимум одним отрезком.

Начинать можно с рассмотрения угловых клеток таблицы, это позволяет провести первые отрезки. Затем возможно рассмотреть клетки по краям таблицы. По мере проведения отрезков между числами, начинает уменьшаться количество возможных вариантов построения новых отрезков. Если к числу приходит необходимое

количество отрезков, значит, оно уже не может соединяться с другими своими соседями.

В условии написано, что сообщение составляет каждая третья буква, но не указано, с какой буквы следует начинать чтение. Выписывая три возможных варианта, получаем, что читаемый текст будет лишь в случае чтения каждой третьей незачёркнутой буквы, начиная с первой.

Ответ: сегодня.

Задача 4 (8-9, 10 классы)

Для зашифрования сообщении каждая его буква заменяется числом по таблице. В результате получается числовая последовательность $x_1, \dots, x_n$. Затем вырабатывают последовательность $\gamma_1, \gamma_2, \dots$ по следующему правилу: γ_1 – некоторое натуральное число, γ_2 – сумма цифр квадрата γ_1, увеличенная на 1, и т.д. Например, если $\gamma_1 = 7$, то $\gamma_2 = 14$, $\gamma_3 = 17$ и т.д. После этого выбирается некоторое натуральное t и формируется зашифрованное сообщение по правилу: $r_{32}(x_1 + \gamma_t), \dots, r_{32}(x_n + \gamma_{t+n-1})$, где $r_{32}(a)$ – остаток от деления числа a на 32. Известно, что для $\gamma_1 = 2019$ и некоторого t получился следующий шифртекст: 10, 6, 26, 22, 15, 13, 20, 13, 29, 13, 28, 23, 4. Восстановите исходное сообщение.

А	Б	В	Г	Д	Е Ё	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Решение:

Будем перебирать возможные значения t, а затем, «раскрутив» последовательность $\gamma_t, \dots, \gamma_{t+n-1}$, попробуем расшифровать на ней текст. Занесем в таблицу

последовательность $\gamma_1, \gamma_2, \ldots$ и соответствующий открытый текст (ОТ), который получается если расшифровать шифртекст с помощью последовательности $\gamma_t, \ldots, \gamma_{t+n-1}$.

t	γ_i	ОТ		ОТ		ОТ		ОТ		ОТ	
1	2019	7	З								
2	28	10	К	14	О						
3	20	6	Ж	18	Т	22	Ц				
4	5	17	С	21	Х	1	Б	5	Е		
5	8	7	З	14	О	18	Т	30	Ю	2	В
6	11	2	В	4	Д	11	Л	15	П	27	Ы
7	5	15	П	8	И	10	К	17	С	21	Х
8	8	5	Е	12	М	5	Е	7	З	14	О
9	11	18	Т	2	В	9	Й	2	В	4	Д
10	5	8	И	24	Ш	8	И	15	П	8	И
11	8	20	Ф	5	Е	21	Х	5	Е	12	М
12	11	12	М	17	С	2	В	18	Т	2	В
13	5	31	Я	18	Т	23	Ч	8	И	24	Ш
14	8			28	Ь	15	П	20	Ф	5	Е
15	11					25	Щ	12	М	17	С
16	5							31	Я	18	Т
17	8									28	Ь

Нетрудно из таблицы заметить, что последовательность $\{\gamma_i\}$ периодическая с периодом (5, 8, 11) и подходом (2019, 28, 20), поэтому для расшифрования сообщения достаточно начинать расшифровывать при $t = 1, \ldots, 6$. . Осмысленный текст получается при $t = 5$.

Ответ: выходим в шесть.

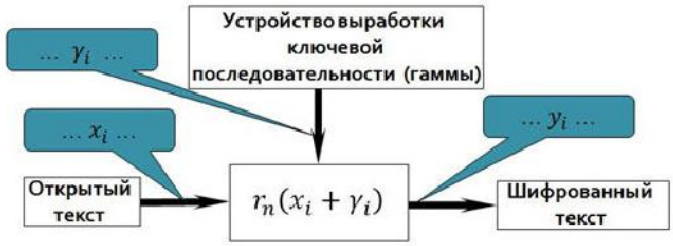
Комментарий

Здесь в качестве способа преобразования открытого сообщения использовался *шифр модульного гаммирования,* устроенный следующим образом. Пусть открытое сообщение $x_1, \dots, x_t$ состоит из символов алфавита $X = \{0, \dots, n-1\}$. Выберем некоторую последовательность (*гамму*) $\gamma_1, \dots, \gamma_t \in X$ и вычислим

$$y_i = r_n(x_i + \gamma_i), i = 1, \dots, t.$$

Сформированная последовательность $y_1, \dots, y_t$ и будет шифртекстом.

Общий принцип работы шифра гаммирования пояснен на рис. ниже:



Стойкость такого шифра существенным образом определяется выбором последовательности $\gamma_1, \dots, \gamma_t$. Эта последовательность должна быть, как говорят, *случайной* и *равновероятной*. Для ее выработки используют так называемые *генераторы псевдослучайных последовательностей* (ГПСЧ). В настоящей задаче гамма не являлась случайной и равновероятной, и, в частности, по этой причине удалось восстановить по шифрованному тексту исходное сообщение.

Задача 5 (8-11 классы)

Для зашифрования осмысленного слова его буквы переводят в числа $x_1, x_2, \dots, x_n$ по таблице. Затем выбирают натуральные числа x_0 и k. Далее число x_0 приписывают в начало последовательности $x_1, x_2, \dots, x_n$, а число $x_{n+1} = x_0 + 19^{n+4}$ (где n – длина слова) – в ее конец. Получившаяся в результате последовательность $x_0, x_1, \dots, x_n, x_{n+1}$ затем преобразуется в последовательность $y_0, y_1, \dots, y_n, y_{n+1}$ по формуле $y_i = r_{32}(x_i + 6x_i \cdot k^3 + k)$, $i = 0,1, \dots, n+1$, где $r_{32}(a)$ – остаток от деления числа a на 32. Затем числа $y_0, y_1, \dots, y_{n+1}$ заменяют буквами согласно таблице. В результате получили вот что: **КЙЫЦНБНЦЛ**. Какое слово было зашифровано?

А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
0	1	2	3	4	5		6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Решение

Нетрудно понять, что длина слова $n = 7$, а также несложно найти остаток $r_{32}(19^{11}) = 11$.

Преобразуем зашифрованный текст в последовательность чисел:

$$y_0 = 10, y_1 = 9, y_2 = 27, y_3 = 22, y_4 = 13, y_5 = 1,$$
$$y_6 = 13, y_7 = 22, y_8 = 11.$$

Из условия следует, что $x_8 - x_0 = 11$. Рассмотрим разность

$$r_{32}(y_8 - y_0) = r_{32}(x_8 + 6x_8 \cdot k^3 + k - x_0 - 6x_0 \cdot k^3 - k) =$$
$$= r_{32}\big((1 + 6k^3) \cdot (x_8 - x_0)\big) = r_{32}\big(11 \cdot (1 + 6k^3)\big).$$

Имеем:

$$r_{32}\big(11 \cdot (1 + 6k^3)\big) = 1.$$

Заметим, что $r_{32}(3 \cdot 11) = 1$. Откуда находим $r_{32}(1 + 6k^3) = 3$. Значит,

$$1 + 6k^3 = 3 + 32t \Leftrightarrow 3k^3 = 1 + 16t \Leftrightarrow$$
$$\Leftrightarrow 33k^3 = 11 + 11 \cdot 16t$$

Значит, $r_{16}(33k^3) = r_{16}(k^3) = 11$. В итоге

$$k^3 = 11 + 16p.$$

При $p = 1$ получим $k^3 = 27$. Отсюда $k = 3$. Опробуем полученное значение. Согласно правилу зашифрования

$$y_1 = 9 = r_{32}(x_1 + 6x_1 \cdot 27 + 3) = r_{32}(x_1 \cdot 3 + 3),$$
$$\Leftrightarrow 3x_1 + 3 = 9 + 32t \Leftrightarrow 3x_1 = 6 + 32t$$

Т.е. $r_{32}(3x_1) = 6 \Longrightarrow r_{32}(x_1) = 2$. Продолжая дальше получим:

$$y_2 = 27 = r_{32}(x_2 + 6x_2 \cdot 27 + 3) = r_{32}(x_2 \cdot 3 + 3),$$
$$\Leftrightarrow 3x_2 + 3 = 27 + 32t \Leftrightarrow 3x_2 = 24 + 32t$$

Т.е. $r_{32}(3x_2) = 24 \Longrightarrow r_{32}(x_2) = 8$. В итоге получим

Ответ: ВИСОКОС.

Комментарий

Если каждому элементу множества X поставлен в соответствие ровно один элемент множества Y так, что при этом любой элемент множества Y окажется сопоставлен ровно одному элементу множества X, то говорят, что между данными множествами установлено *взаимно-однозначное, или биективное, соответствие (отображение)*. Биективное отображение множества X в себя называется *подстановкой элементов множества X*.

Идея предложенной задачи заключается в следующем наблюдении. Если целые числа a, b – нечетные, c – четное, $m, k, l \in \mathbb{N}, m > 1$, то функция

$$f(x, y) = r_{2^m}(a \cdot x + c \cdot x^k \cdot y^l + b \cdot y)$$

будет задавать подстановку по каждой переменной на множестве $\{0,1,\dots,2^m-1\}$ – остатков от деления на 2^m. Сказанное означает, что для любого фиксированного $x \in \{0,1,\dots,2^m-1\}$ функция $g(y)=f(x,y)$ задает подстановку элементов множества $\{0,1,\dots,2^m-1\}$. Аналогичное справедливо при любом фиксированном $y \in \{0,1,\dots,2^m-1\}$.

Данный факт имеет наглядную интерпретацию. Если записать таблицу значений указанной функции $f(x,y)$,

$x\backslash y$	0	1	…	j	…	2^m-1
0	$f(0,0)$	$f(0,1)$	…	$f(0,j)$	…	$f(0,2^m-1)$
1	$f(1,0)$	$f(1,1)$	…	$f(1,j)$	…	$f(1,2^m-1)$
…	…	…	…	…	…	…
i	$f(i,0)$	$f(i,1)$	…	$f(i,j)$	…	$f(i,2^m-1)$
…			…		…	
2^m-1	…	…	…	…	…	…

то можно заметить, что в каждой ее строке (и в каждом ее столбце) присутствуют все элементы $\{0,1,\dots,2^m-1\}$ при этом в точности по одному разу. Таблицы, обладающие указанным свойством, называются в математике *латинскими квадратами (латинскими прямоугольниками)*. Впервые латинские квадраты (4-го порядка) были опубликованы в книге «Шамс аль Маариф» («Книга о Солнце Гнозиса»), написанной Ахмадом аль-Буни в Египте приблизительно в 1200 г. Свое название латинские квадраты получили благодаря математику Леонарду Эйлеру, который вместо чисел в подобные таблицы записывал буквы латинского алфавита.

Латинские квадраты находят свое применение в криптографии. В частности предложенный в задаче способ зашифрования сообщения основан на использовании как раз латинского квадрата. Данный способ называют *табличным гаммированием.*

Задача 6 (8-11 классы)

Каждому из четырех абонентов A_1, A_2, A_3, A_4 надо выдать по два уравнения вида $aw + bx + cy + +dz = t$, где $a, b, c, d, t, w, x, y, z \in \{0,1\}$. Значения секретных битов w, x, y, z одинаковы для всех абонентов и им заранее неизвестны. Приведите хотя бы один пример уравнений, которые надо выдать этим четырем абонентам, чтобы каждая пара $\{A_1, A_3\}, \{A_1, A_4\}, \{A_2, A_3\}$ могла достоверно вычислить w, x, y, z, но чтобы при этом: 1) ни одна другая пара абонентов не могла бы достоверно вычислить более одного секретного бита; 2) ни один абонент в одиночку не был в состоянии достоверно вычислить даже один секретный бит. Например, если абонент A_1 получит уравнения $\{w + x + y + z = 1;\ w + x + 0 \cdot y + 0 \cdot z = 1\}$, а A_2 – $\{w + 0 \cdot x + y + 0 \cdot z = 0;\ w + x + 0 \cdot y + +z = 0\}$. Тогда, объединившись, из имеющихся в их распоряжении четырех уравнений они однозначно найдут, что $w = 1, x = 0, y = 1, z = 1$. При этом будем говорить, что пара абонентов $\{A_1, A_2\}$ может *достоверно вычислить* секретные биты w, x, y, z. Здесь традиционно полагается, что 1+1=0.

Решение

Пусть w_0, x_0, y_0, z_0 – значения секретных битов w, x, y, z. Решим прежде задачу, предполагая, что все

секретные биты равны нулю: $w_0 = x_0 = y_0 = z_0 = 0$. Затем в уравнениях можно будет сделать замену $w \to w + w_0, \dots, z \to z + z_0$ и тем самым получить решение задачи в общем случае.

Запишем теперь какую-нибудь систему из четырех уравнений, которой удовлетворяют *только* нулевые значения. Например,

$$w + x = 0 \quad (1) \qquad y + z = 0 \quad (3)$$
$$x + y = 0 \quad (2) \qquad w + x + y = 0 \quad (4)$$

Запишем еще одно уравнение, сложив эти четыре:

$$x + y + z = 0 \quad (5)$$

Система из *любых* четырех уравнений из набора (1) – (5) имеет только нулевое решение.

Далее идея в следующем. Если пара абонентов должна уметь находить все биты, то этой паре выдадим четыре *различные* уравнения из набора (1) – (5), если же нет, то хоть одно уравнение у этой пары должно быть общим.

Замечание. *Здесь нет четких алгоритмов и успех заранее не гарантирован. Возможно, следовало выбрать какие-то другие уравнения (1) – (4). Заметим, например, что абонентам, которые не должны уметь находить секрет, нельзя выдать уравнения (1), (2) и (4), так как значение бита z они не найдут, но определят, что* $w = x = y = 0$, *а это по условию недопустимо. Никакому абоненту нельзя выдать уравнения (2) и (5), так как из них следует, что* $z = 0$.

Абонентам раздать уравнения можно так: $A_1: (1), (2)$; $A_2: (1), (5)$; $A_3: (3), (4)$; $A_4: (4), (5)$.

Выполнив замену, запишем ответ в общем случае.

Ответ: Например,

$$A_1: w + x = w_0 + x_0,\ x + y = x_0 + y_0;\ A_2: w + x = w_0 + x_0,\ x + y + z = x_0 + y_0 + z_0;$$
$$A_3: y + z = y_0 + z_0,\ w + x + y = w_0 + x_0 + y_0;$$
$$A_4: w + x + y = w_0 + x_0 + y_0,\ x + y + z = x_0 + y_0 + z_0.$$

Комментарий

В данной задаче описывается более общий случай разделения секрета по сравнению с пороговой схемой Шамира, при которой доступ к секрету получают определенные группы абонентов, которых называют *уполномоченными*. Здесь в точности задаются конкретные абоненты, которые могут получить секрет (вообще говоря соответствующие группы могут различаться по числу участников). Такие схемы разделения секрета называют *структурами доступа*.

Существует общий незамысловатый алгоритм построения структуры доступа для разделения одного бита секрета между заданными группами абонентов с помощью дополнительных битов *забеливания*, которые складываются с секретным битом, при этом для каждой группы используются разные забеливающие биты. Однако этот подход не позволяет получать эффективные структуры доступа. Под понятием *информационная эффективность* структуры доступа понимается отношение числа распределенных секретных битов в структуре доступа к длине самой длинной доли (числу выданных бит абоненту).

Можно доказать, что эффективность не превосходит 1. Задача построения таких структур доступа до сих пор в общем случае не решена.

Сама постановка в предложенной для решения задаче несколько усложняется: необходимо распределить 3 секретных бита (для 8-10 классов) или 4 секретных бита (для 11 класса) между 4-мя абонентами и выдать при этом каждому абоненту не более 2-ух бит. То есть эффективность такой структуры доступа будет больше 1 ($3/2$ в задаче для 8-10 классов и $4/2$ в задачах для 11 класса)! Здесь нет обмана – просто в предложенной задаче разрешено не уполномоченным группам абонентов получать суммы секретных бит, а в задаче для 11 класса – еще и один секретный бит при условии, что остальные 3 бита по-прежнему должны для них оставаться в секрете.

Задача 7 (11 класс)

Саша решил отправить Маше записку. Для этого каждую букву сообщения он заменил комбинацией из 0 и 1 согласно таблице (А – 00000, Б – 00001, …, Я – 11111). Взяв день "Д" и номер месяца "М" своего рождения Саша вычислил $u_1 = Д^2 + М^2, u_2 = Д \cdot М, u_3 = Д - М$. Далее Саша вычислил четвертое $u_4 = r_{32}(u_1 + u_2 u_3)$, пятое $u_5 = r_{32}(u_2 + u_3 u_4)$, …, n-ое число $u_n = r_{32}(u_{n-3} + +u_{n-2} u_{n-1})$, где $r_{32}(a)$ – остаток от деления числа a на 32. К i-му биту символу исходного сообщения (0 или 1) он прибавил число u_i и взял остаток от деления на 2. Полученную последовательность из 0 и 1 он вновь преобразовал в буквы по таблице и получил следующее

сообщение: **ЖДУЛЩБШЛТВШЦЧ**. Помогите Маше прочитать его.

А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Решение

По условию числа u_k прибавляются к битам открытого текста, а результат заменяется остатком от деления на 2 (то есть на 0 или 1). Поэтому сразу заменим u_k его остатком от деления на 2: считаем, что $u_k = 0$ (если изначально u_k было четным) или $u_k = 1$ (если оно было нечетным). Вычисление остатка от деления на 32 при построении последовательности $u_1, u_2, \ldots$ никакой роли не играет (четные числа дают четный остаток, а нечетные – нечетный).

Оказывается, в зависимости от четности чисел Д, М могут быть получены всего три различные последовательности $u_1, u_2, \ldots$, а именно:

1. Числа Д, М нечетные. Тогда $u_1 = 0, u_2 = 1, u_3 = 0, \ldots$
2. Числа Д, М имеют разную четность. Тогда $u_1 = 1, u_2 = 0, u_3 = 1, \ldots$
3. Числа Д, М четные. Тогда $u_1 = u_2 = \cdots = u_{32} = 0$. В этом случае текст Машиной записки остался бы без изменения, что, очевидно, не так.

Далее необходимо в первых двух случаях вычислить последовательность $\{u_n\}$ полностью, вычесть ее из

зашифрованного текста (**ЗТ**) и убедиться, что читаемый вариант получается во втором случае (см. таблицу).

	Ж	Д	У	Л	Щ	Б	Ш	Л	У	В	Ш	Ц	Ч
	00110	00100	10011	01011	11001	00001	11000	01011	10011	00010	11000	10110	10111
Д , М **нечетные**													
u_n	01001	00100	10010	01001	00100	10010	01001	00100	10000	01001	00100	10010	01001
ЗТ -u_n	01111	00000	00001	00010	11101	10011	10001	01111	00011	01011	11100	00100	11110
	П	А	Б	В	Э	У	С	П	Г	Л	Ь	Д	Ю
Д , М **разной четности**													
u_n	10111	01110	11101	11011	10111	01110	11101	11011	10110	01110	11101	11011	10111
ЗТ -u_n	10001	01010	01110	10000	01110	01111	00101	10000	00101	01100	00101	01101	00000
	С	К	О	Р	О	П	Е	Р	Е	М	Е	Н	А

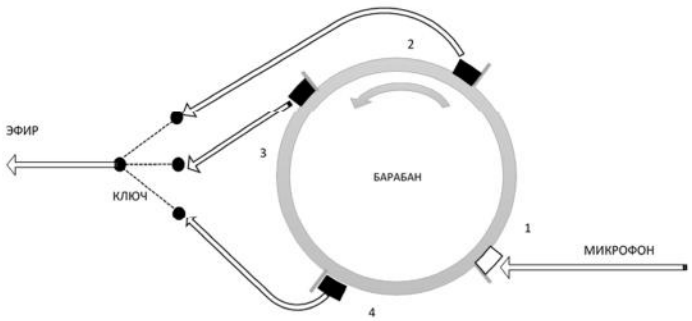
Ответ: **СКОРОПЕРЕМЕНА**

Комментарий

Представленная задача также относится к шифрам гаммирования, представленным ранее в задаче 4.

Задача 8 (11 класс)

Звук записывается на магнитный слой барабана (см. рис.), который вращается с постоянной скоростью, совершая один оборот за 4 секунды. Рядом с барабаном по окружности через равные расстояния размещены записывающая (1) и три читающие головки (2), (3), (4). В каждый момент времени в телефонную линию передается сигнал с одной из читающих головок. Устройство спроектировано так, что каждый участок сигнала будет передан в линию один раз, а сама передача стартует, как



только начало записи окажется у 3-й читающей головки.Сколько различных вариантов звука, переданного в линию, может получиться, если сообщение длилось 20 секунд?

Решение

Решим задачу в общем случае, когда передача длилась n секунд. Так как переключение между читающими головками происходит раз в секунду, весь звук можно разбить на n фрагментов по 1 секунде и тогда звук, переданный в линию, будет перестановкой этих фрагментов. Обозначис количество возможных перестановок $T(n)$.

Представим весь процесс в виде таблицы, элементами которой являются номера фрагментов. Например, на второй секунде, с которой начинается передача, на пишущей головке будет 3-ий фрагмент звука, 2-ой фрагмент будет на (2)-ой читающей головке, а 1-ый фрагмент на (3)-ей читающей головке. Передача закончится на $n + 1$ секунде.

Сек.	Пишущая головка	Читающая головка			В линию передан
		2)	3)	4)	
0	1	–	–	–	–
1	2	1	–	–	–
2	3	2	1	–	2 или 1
3	4	3	2	1	3, 2 или 1
4	5	4	3	2	4, 3 или 2
...	...	...	...	...	
$n-1$	n	$n-1$	$n-2$	$n-3$	
n	–	n	$n-1$	$n-2$	
$n+1$	–	–	n	$n-1$	n или $n-1$

На $n+1$ секунде в линию может быть передан n или $n-1$ фрагмент звука. По очереди рассмотрим оба случая.

1. Пусть на $n+1$ секунде в линию был передан -ый фрагмент (см. таблицу). Тогда -ый фрагмент не мог быть передан на предыдущей секунде. Если посмотреть на таблицу то видно, что количество перестановок фрагментов в этом случае совпадает с $T(n-1)$, то есть количеством способов переставить звук длины $n-1$.

Читающая головка			В линию
(2)	(3)	(4)	
2	1	–	2 или 1
3	2	1	3, 2 или 1
4	3	2	4, 3 или 2
…	…	…	
$n-1$	$n-2$	$n-3$	
n	$n-1$	$n-2$	$n-1$ или $n-2$
–	n	$n-1$	n

2. Пусть на $n+1$ секунде в линию был передан $(n-1)$-ый фрагмент (см. таблицу). Тогда $(n-1)$-ый фрагмент не мог быть передан на предыдущих секундах. Так как n-ый фрагмент должен уйти в линию, то он должен быть передан

Читающая головка			В линию
(2)	(3)	(4)	
2	1	–	2 или 1
3	2	1	3, 2 или 1
4	3	2	4, 3 или 2
…	…	…	
$n-1$	$n-2$	$n-3$	$n-2$ или $n-3$
n	$n-1$	$n-2$	n
–	n	$n-1$	$n-1$

в момент времени n. Тогда до $(n-1)$-ой должно быть

передано $(n-2)$ последовательных фрагментов, что может быть сделано $T(n-2)$ способами.

Таким образом $T(n) = T(n-1) + T(n-2)$. Тогда для нахождения количества перестановок $T(n)$ для любого n, достаточно найти $T(1), T(2)$.

Читающая головка			В линию
2)	3)	4)	
–	1	–	1

$T(1) = 1$

Читающая головка			В линию
2)	3)	4)	
2	1	–	2 или 1
	2	1	2 или 1

$T(2) = 2$

Остается с использованием формулы $T(n) = T(n-1) + T(n-2)$ вычислить нужное значение.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	2	3	5	8	13	21	34	55	89	144	233	377	610	987	1597	2584	4181	6765	10946

Ответ: 10946

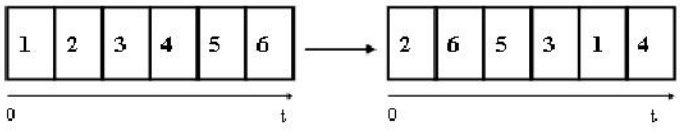
Комментарий

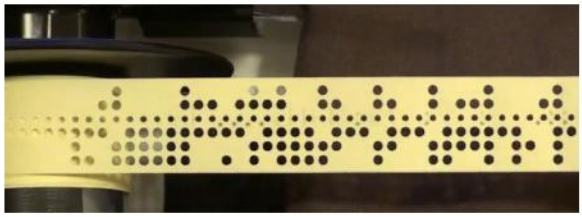
Развитие средств связи к началу Великой Отечественной войны обусловило необходимость перехода от медленных систем предварительного шифрования текстовой информации к синхронному линейному засекречиванию телефонных переговоров непосредственно в процессе связи. В этих условиях стало актуальным создание особой секретной телефонии, которая обеспечила бы надёжную защиту содержания перехваченного противником телефонного разговора.

Задача была решена коллективом специалистов под руководством начальника лаборатории и главного инженера по радио в Центральном научно-исследовательском институте связи Народного комиссариата почт и телеграфов (НИИС НКПиТ) **Владимира Александровича Котельникова**. В сложнейших условиях военного времени в лаборатории был разработан и построен принципиально новый телефонный шифратор «мозаичного» типа «Соболь-П», не имевший аналогов в мире.

Шифратор осуществлял частотно-временные («мозаичные») преобразования речевого сигнала с последующей перестановкой отрезков полученного сигнала по времени. Частотные преобразования заключались в делении сигнала на три-четыре частотных поддиапазона посредством системы полосовых фильтров. К некоторым из частотных поддиапазонов применялись частотные инверсии. Далее поддиапазоны случайным образом перемешивались.

Сигнал, полученный после частотных преобразований, делился на отрезки по 100 миллисекунд и задерживался для осуществления перемешивания отрезков во времени посредством записи на магнитный барабан. Перемешивание частотно-временных отрезков осуществлялось с помощью специального узла, управлявшегося посредством перфоленты, отверстия в которой проделывались случайным (непредсказуемым) образом.





Описанные преобразования надёжно защищали содержание подлежащего засекречиванию речевого сообщения не только от непосредственного подслушивания сигнала в канале связи, но и от попыток восстановить исходную речь доступными на тот момент техническими средствами и методами.

Уровень безопасности телефонных переговоров, обеспечиваемый аппаратурой «Соболь-П», для своего времени являлся беспрецедентным. По каналам связи, оборудованным аппаратурой «Соболь-П», разрешалась передача совершенно секретных донесений и приказов.

Задача 9 (11 класс)

Рассмотрим девять чисел $k_1, \dots, k_9$, где $k_i \in \{0,1,2\}$. При этом хотя бы одно число k_i отлично от нуля. С помощью этих чисел вырабатывают последовательность $u_1, u_2, \dots, u_{2019}$ по формулам: $u_1 = k_1,\ u_2 = k_2, \dots,\ u_9 = k_9$, $u_{i+9} = r_3(u_i + u_{i+1}),\ i = 1,2, \dots, 2010$, где $r_3(a)$ – остаток от деления числа a на 3. Найдите такое наименьшее натуральное число l, что какие бы исходные числа $k_1, \dots, k_9$ мы ни взяли, в последовательности $u_1, u_2, \dots, u_l$ каждое из чисел 0, 1 и 2 гарантированно встретится хотя бы один раз.

Решение

Для каждого набора $\boldsymbol{k} = (k_1, \dots, k_9)$ укажем такое минимальное l, что в соответствующей последовательности $u_1, u_2, \dots, u_l$ присутствует каждое из чисел 0, 1 и 2. Затем среди всех таких l останется выбрать наибольшее – это и будет ответом в задаче.

1. В наборе $\boldsymbol{k}$ встречается каждое из чисел 0, 1 и 2. Тогда искомое l не превосходит 9;

2. Набор $\boldsymbol{k}$ состоит только из 1. Тогда $u_{10} = \dots = u_{17} = 2$ и $u_{18} = 0$. Значит $l = 18$;

3. В наборе $\boldsymbol{k}$ присутствуют и 1, и 2, но нет 0. Значит среди чисел $u_1, u_2, \dots, u_9$ есть два соседних (u_s и u_{s+1}), одно из которых равно 1, а другое 2. Тогда $u_{s+9} = 0$ и $l \leq 17$;

4. Набор $\boldsymbol{k}$ состоит из 0 и 1. Число 2 впоследствии дадут только две рядом стоящие 1. Поэтому рассмотрим варианты:

a) в $\boldsymbol{k}$ есть рядом стоящие 1. Тогда $l < 19$;

b) в $\boldsymbol{k}$ нет рядом стоящих 1. Здесь возможны следующие случаи:

- Есть хоть одна 1 «не с краю». То есть найдется номер s такой, что $2 \leq s \leq 8$ и $k_s = 1$. Рядом стоящих 1 нет, поэтому $k_{s-1} = k_{s+1} = 0$. Тогда $u_{s+8} = u_{s+9} = 1$. Следовательно, $u_{s+17} = 2$ и $l \leq 25$;
- 1 есть только «с краю». Пусть $\boldsymbol{k} = (1,0, \dots, 0)$. В этом случае начало последовательности $u_1, u_2, \dots$ можно вычислить непосредственно: $\{u_n\} = \{1,$ 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,

0, 0, 0, 1, 2, 1, 0,…} и убедиться, что $l = 27$. Пусть $\boldsymbol{k} = (1,0,\dots 0,1)$. Тогда $\{u_n\}$ ={1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 2, 1, 0…} и $l = 18$. И, наконец, для $\boldsymbol{k} = (0,\dots 0,1)$ находим $\{u_n\}$ ={0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 2, 1, 0, …}, $l = 26$.

Отметим, что случаи, «$\boldsymbol{k}$ состоит только из 2» и «$\boldsymbol{k}$ состоит только из 0 и 2» эквивалентны случаям 2 и 4 соответственно. Действительно, если в последовательности $\{u_n\}$, отвечающей набору $2 \cdot \boldsymbol{k}$, заменить все 2 на 1, а 1 на 2, то получится последовательность, соответствующая набору $\boldsymbol{k}$.

Ответ: $l = 27$.

Комментарий

Последовательность чисел $\{u_n\}$, удовлетворяющая при некоторых числах $m \in \mathbb{N}, a_0, \dots, a_{m-1} \in \mathbb{R}$ рекуррентному соотношению при всех $n \in \mathbb{N}_0$

$$u_{n+m} = a_{m-1}u_{n+m-1} + \cdots + a_1u_{n+1} + a_0u_0,$$

называется линейной рекуррентной последовательностью (ЛРП) порядка m, а многочлен

$$f(x) = x^m - a_{m-1}x^{m-1} - \cdots - a_1x - a_0$$

– ее характеристическим многочленом. ЛРП – это такая рекуррентная последовательность, в которой очередной член есть фиксированная линейная комбинация ее предыдущих членов. Примерами ЛРП служат *последовательности Фибоначчи, арифметические и геометрические прогрессии.*

ЛРП применяются для построения генераторов псевдослучайных последовательностей, которые в свою

очередь используются в *шифрах гаммирования* в качестве источника случайной и равновероятной *гаммы*. Стоит отметить, что для того, чтобы ЛРП могла быть использована в таких генераторах, она должна удовлетворять достаточно большому набору криптографических характеристик, в противном случае гамма может оказаться «плохой» и соответствующая шифрсистема окажется нестойкой. Параметр l можно рассматривать как одну из таких характеристик. Он показывает наименьшую длину отрезка последовательности, на которой каждое из чисел 0, 1, 2 встретиться хотя бы один раз. Отсутствие хотя бы одного из этих чисел в отрезке последовательности заведомо означало бы неравновероятность рассматриваемой ЛРП $\{u_n\}$.

ПРИЛОЖЕНИЕ
(основные обозначения и свойства остатков)

Обозначения:

- запись $A \Leftrightarrow B$ означает, что утверждение A выполняется тогда и только тогда, когда выполняется утверждение B, а $A \Rightarrow B$ означает, что из A следует B;
- запись $a \in M$ означает принадлежность элемента a множеству M;
- $\text{НОД}(a, b)$ – наибольший общий делитель чисел a и b;
- $k = 1,2,3 \ldots$ – индекс k пробегает (принимает последовательно) значения $1,2,3 \ldots$;
- $\mathbb{N}$ – множество натуральных чисел;
- $r_n(x)$ – остаток от деления натурального числа x на ненулевое целое число n.

Определение. *Разделить целое число a на ненулевое целое число n с остатком означает найти такие целые числа q, r такие, что выполнено равенство $a = q \cdot n + r$, и при этом $0 \le r < |n|$ – остаток от деления числа a на n, который обозначают как $r_n(a)$, а число q называют неполным частным.*

Пример. Остаток от деления 7 на 3 равен 1 (то есть $r_3(7) = 1$), поскольку $7 = 3 \cdot 2 + 1$. В то же время остаток от деления –2 на 3 так же равен 1 ($r_3(-2) = 1$): $-2 = 3 \cdot (-1) + 1$.

Теорема. *Любое целое число a можно разделить с остатком на ненулевое целое число n, при этом остаток и неполное частное определены однозначно.*

Утверждение. *Справедливы следующие свойства:*

1. $r_n(a + c) = r_n(r_n(a) + r_n(c))$;

2. $r_n(a \cdot c) = r_n(r_n(a) \cdot r_n(c))$;

3. $r_n(a) = r_n(c) \Leftrightarrow a - c$ *делится на* n.

Проверка работ проводилась централизовано по единым критериям. Всего дипломами I, II, III степени награждены более 150 участников. Задания олимпиады были подготовлены для каждой возрастной категории (8-9, 10 и 11 классы) в нескольких равноценных вариантах. В сборнике приводились условия и решения одной из задач каждого типа.

С задачами прошедших олимпиад по математике и криптографии и их решениями можно ознакомиться:

- на сайте www.cryptolymp.ru в разделе «Подготовка к олимпиаде» и «Архив задач»;
- на сайте Академии ФСБ России по адресу www.academy.fsb.ru (раздел для абитуриентов);
- в учебно-методическом журнале «Математика», Издательский дом «Первое сентября» (ежегодно в одном из апрельских выпусков, www.1september.ru);
- в книге «Введение в криптографию» (М.:МЦНМО, 2012);
- в книге «Олимпиады по криптографии и математике для школьников» (М.: МЦНМО, 2013);
- также можно получить доступ к системе дистанционного обучения для подготовки к олимпиаде на сайте: http://www.v-olymp.ru.