МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7 НА ТЕМУ:**

**Исследование блочных шифров**

Выполнила студентка 3 курса 4 группы

Коржова В. С.

Минск 2023

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

**Теоретические сведения**

Блочное зашифрование (расшифрование) предполагает разбиение исходного открытого (зашифрованного) текста на равные блоки, к которым применяется однотипная процедура зашифрования (расшифрования). Указанная однотипность характеризуется прежде всего тем, что процедура зашифрования (расшифрования) состоит из совокупности повторяющихся наборов преобразований, называемых раундами.

**Алгоритм DES** строится на основе сети Фейстеля. Входной блок данных, состоящий из 64 битов, преобразуется в выходной блок идентичной длины. В алгоритме широко используются рассеивания (подстановки) и перестановки битов текста, о которых мы упоминали выше. Комбинация двух указанных методов преобразования образует фундаментальный строительный блок DES, называемый раундом или циклом. Общая схема алгоритма DES представлена на рисунке 1.

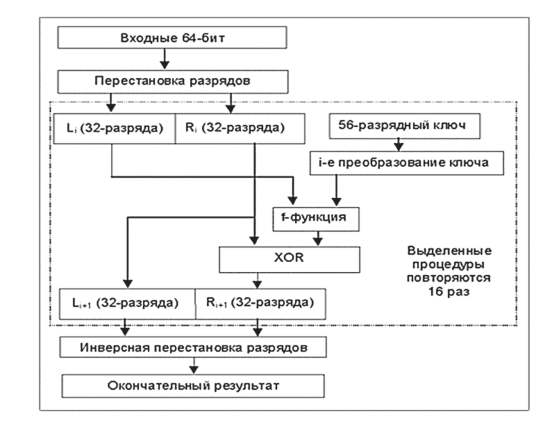


Рисунок 2 – общая схема алгоритма DES

Первоначальная перестановка разрядов (*IP*) производится при помощи специальной таблицы, по которой определяется позиция бита по указанному индексу. В реализованном по практическому заданию приложении для этого используется список *ip*, имеющий следующий вид:

    \_\_ip = [57, 49, 41, 33, 25, 17, 9, 1,

        59, 51, 43, 35, 27, 19, 11, 3,

        61, 53, 45, 37, 29, 21, 13, 5,

        63, 55, 47, 39, 31, 23, 15, 7,

        56, 48, 40, 32, 24, 16, 8, 0,

        58, 50, 42, 34, 26, 18, 10, 2,

        60, 52, 44, 36, 28, 20, 12, 4,

        62, 54, 46, 38, 30, 22, 14, 6

    ]

Листинг 1 – Первоначальная перестановка

Например, мы хотим переставить 4 бит входного блока. Индексация с 0, соответственно индекс данного бита 3. По данному списку заметно, что бит будет размещен на 16-й позиции. Аналогично, бит с индексом 57 будет размещен на первой, с индексом 49 – на второй и т.д.

Один блок данных подвергается преобразованию (и при зашифровании, и при расшифровании) в течение 16 раундов. После первоначальной перестановки. 64-битный блок данных делится на правую (*R*0) и левую (*L*0) половины длиной по 32 бита выполняются 16 раундов одинаковых действий.

Схема одного раунда DES представлена на рисунке 2.

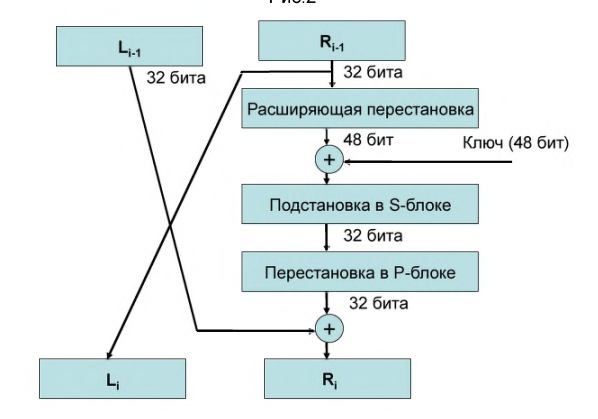


Рисунок 2 – один раунд DES

Вначале правая часть блока *Ri* расширяется до 48 битов с использованием таблицы, которая определяет перестановку плюс расширение на 16 битов. Эта операция приводит размер правой половины в соответствие с размером ключа для выполнения операции XOR. В реализованном по практическому заданию приложении для этого используется список *expansion\_table*, имеющий следующий вид:

    \_\_expansion\_table = [

        31,  0,  1,  2,  3,  4,

         3,  4,  5,  6,  7,  8,

         7,  8,  9, 10, 11, 12,

        11, 12, 13, 14, 15, 16,

        15, 16, 17, 18, 19, 20,

        19, 20, 21, 22, 23, 24,

        23, 24, 25, 26, 27, 28,

        27, 28, 29, 30, 31,  0

    ]

Листинг 2 – Таблица расширения

Расширение происходит за счет дублирования элементов. Можно заметить подряд идущие индексы 3, 4, 3, 4.

Затем полученное 48-битное значение подается на вход блока подстановки S, результатом которой является 32-битное значение. Подстановка выполняется в восьми блоках подстановки или восьми *S*-блоках, имеющих следующий вид:

\_\_sbox = [

        # S1

        [14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7,

         0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8,

         4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0,

         15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13],

        # S2

        [15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10,

         3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5,

         0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15,

         13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9],

        # S3

        [10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8,

         13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1,

         13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7,

         1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12],

        # S4

        [7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15,

         13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9,

         10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4,

         3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14],

        # S5

        [2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9,

         14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6,

         4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14,

         11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3],

        # S6

        [12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11,

         10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8,

         9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6,

         4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13],

        # S7

        [4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1,

         13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6,

         1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2,

         6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12],

        # S8

        [13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7,

         1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2,

         7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8,

         2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11],

    ]

Листинг 3 – S-блоки

При выполнении этой операции 48 битов данных делятся на восемь 6-битных подблоков, каждый из которых по соответствующей таблице замен замещается четырьмя битами.

Метод заключается в следующем: первый и последний разряды двоичного числа определяют десятичное число от 0 до 3 (или от 00 до 11 двоичного числа). Пусть это число будет *i*. Средние 4 разряда представляют собой десятичное число от 0 до 15 (двоичное от 0000 до 1111). Пусть это число будет *j*. В таблице ищем число в строке *i* и столбце *j*. Его двоичное представление и является результатом функции.

Например, дано число *B* = 101111, определим *B*’с помощью блока *S*1. *i* = 11 = 3, *b* = 0111 = 7, значение *B*’ = 7 = 0111.

Подстановка с помощью *S*-блоков является одним из важнейших этапов DES. Таблицы замен для этой операции специально спроектированы так, чтобы обеспечивать максимальную криптостойкость. В результате выполнения этого этапа получаются восемь 4-битных блоков, которые вновь объединяются в единое 32-битное значение.

Далее полученное 32-битное значение обрабатывается с помощью перестановки *Р*, которая не зависит от используемого ключа. Таблица перестановки *P* имеет следующий вид:

    \_\_p = [

        15, 6, 19, 20, 28, 11,

        27, 16, 0, 14, 22, 25,

        4, 17, 30, 9, 1, 7,

        23,13, 31, 26, 2, 8,

        18, 12, 29, 5, 21, 10,

        3, 24

    ]

Листинг 4 – Таблица перестановки *P*

Целью перестановки является такое максимальное переупорядочивание битов, чтобы в следующем раунде шифрования каждый бит с большой вероятностью обрабатывался другим S-блоком.

И наконец, результат перестановки объединяется с помощью операции XOR с левой половиной первоначального 64-битного блока данных. Затем левая и правая половины меняются местами, и начинается следующий раунд.

После выполнения 16-раундового зашифрования 64-битного блока данных осуществляется конечная перестановка (*IP*−1). Она является обратной к перестановке *IP* и имеет следующий вид:

    \_\_fp = [

        39,  7, 47, 15, 55, 23, 63, 31,

        38,  6, 46, 14, 54, 22, 62, 30,

        37,  5, 45, 13, 53, 21, 61, 29,

        36,  4, 44, 12, 52, 20, 60, 28,

        35,  3, 43, 11, 51, 19, 59, 27,

        34,  2, 42, 10, 50, 18, 58, 26,

        33,  1, 41,  9, 49, 17, 57, 25,

        32,  0, 40,  8, 48, 16, 56, 24

    ]

Листинг 5 – Таблица конечной перестановки *IP*−1

Каждый 8-й бит исходного 64-битного ключа отбрасывается. Эти 8 битов, находящих в позициях 8, 16, 24, 32, 40, 48, 56, 64, изначально добавляются в исходный ключ таким образом, чтобы каждый байт содержал четное число единиц. Это используется для обнаружения ошибок при обмене и хранении ключей по известным алгоритмам избыточного кодирования. Один избыточный бит в ключе DES формируется, как видим, в соответствии с кодом простой четности. Этот код позволяет в кодовом слове (в нашем случае – в каждом байте ключа) обнаруживать ошибки, количество которых нечетно.

При расшифровании на вход алгоритма подается зашифрованный текст. Единственное отличие состоит в обратном порядке использования частичных ключей *Ki*. Ключ *K*16 используется в первом раунде, *K*1 – в последнем.

**Практическое задание**

1. Разработать авторское приложение в соответствии с целью лабораторной работы. При этом можно воспользоваться готовыми библиотеками либо программными кодами, реализующими некоторые блочные алгоритмы.

Приложение должно реализовывать следующие операции:

• разделение входного потока данных на блоки требуемой длины с необходимым дополнением последнего блока;

Данное требование реализовано с помощью параметра *padmode=PAD\_PKCS5* функции *triple\_des*.

alg = triple\_des(key\_string, padmode=PAD\_PKCS5)

Листинг 6 – Функция triple\_des

• выполнение требуемых преобразований ключевой информации;

• выполнение операций зашифрования/расшифрования;

• оценка скорости выполнения операций зашифрования/расшифрования;

Для измерения скорости операций зашифрования/расшифрования был использован модуль *python datetime*. Для оценки скорости возьмем исходное сообщение длиной 5000 символов, время выполнения операций представлено на рисунке 3.



Рисунок 3 – Время зашифрования/расшифрования

После выполнения анализа, можно сделать вывод, что на выполнение шифрования уходит больше времени, нежели на выполнения операции дешифрования.

• пошаговый анализ лавинного эффекта с подсчетом количества изменяющихся символов по отношению к исходному слову.

**Эффект лавины** означает, что изменение значения малого количества битов во входном тексте или ключе ведет к «лавинному» изменению значений выходных битов шифротекста. Другими словами, это зависимость всех (или хотя бы половины) выходных битов от каждого входного бита.

В приложении реализована замена 1 случайного символа на случайный символ английского алфавита.

random\_index = random.randint(0, len(message)-1)

random\_letter = message[random\_index]

modified\_letter = random.choice([chr(i) for i in range(97, 123)]).encode()

modified\_message = message[:random\_index] + modified\_letter + message[random\_index+1:]

Листинг 7 – Замена символа на случайны

Результат применения этого кода и последующего шифрования полученного сообщения представлен на рисунке 4.

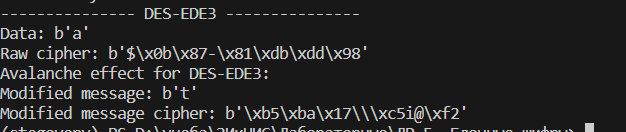


Рисунок 4 – Результат кодирования букв «a» и «t»

В данном случае мы используем входное сообщение, состоящее из 1 буквы – «a». Сообщение дополняется до 8 бит и случайным образом буква заменяется на другую.

Несмотря на то, что символы отличаются лишь на единицу, а все последующие символы совпадают, сообщение после шифрования в первом и втором случаях абсолютно разные, что является доказательством того, что этот эффект является желательным свойством криптографических алгоритмов, так как он делает невозможным прогнозирование выхода алгоритма для незначительно измененных входных данных.

Исследуемый метод шифрования и ключевая информация – в соответствии с вариантом из таблицы 1.

Таблица 1­– Задание по варианту

|  |  |  |
| --- | --- | --- |
| Вариант | Алгоритм | Ключ |
| 3 | DES-EDE3 | По указанию преподавателя |

Алгоритм DES-EDE3: 3DES операции шифрование-расшифрование-шифрование с разными ключами. Для осуществления данного алгоритма в коде используется ключ, длиной 24 байта, который делится на 3 части.

1. Проанализировать влияние слабых ключей и полуслабых ключей на конечный результат зашифрования и на лавинный эффект.

Из-за того, что первоначальный ключ изменяется при получении подключа для каждого раунда алгоритма, определенные первоначальные ключи являются слабыми. Если все биты каждой половины равны 0 или 1, то для всех раундов алгоритма используется один и тот же ключ. Это может произойти, если ключ состоит из одних 1, из одних 0, или если одна половина ключа состоит из одних 1, а другая – из одних 0.

Результат использования слабых ключей на исходном сообщении представлен на рисунке 5.

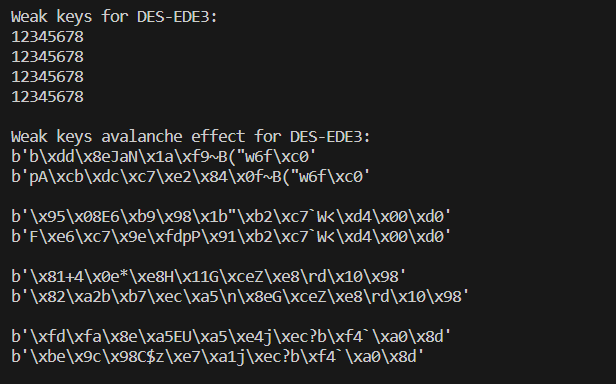


Рисунок 5 – Использование слабых ключей

Кроме того, существую полуслабые ключи ­– некоторые пары ключей при зашифровании переводят открытый текст в идентичный шифртекст. Иными словами, один из ключей пары может расшифровать сообщения, зашифрованные другим ключом пары.

Результат использования полуслабых ключей на исходном сообщении представлен на рисунке 6.

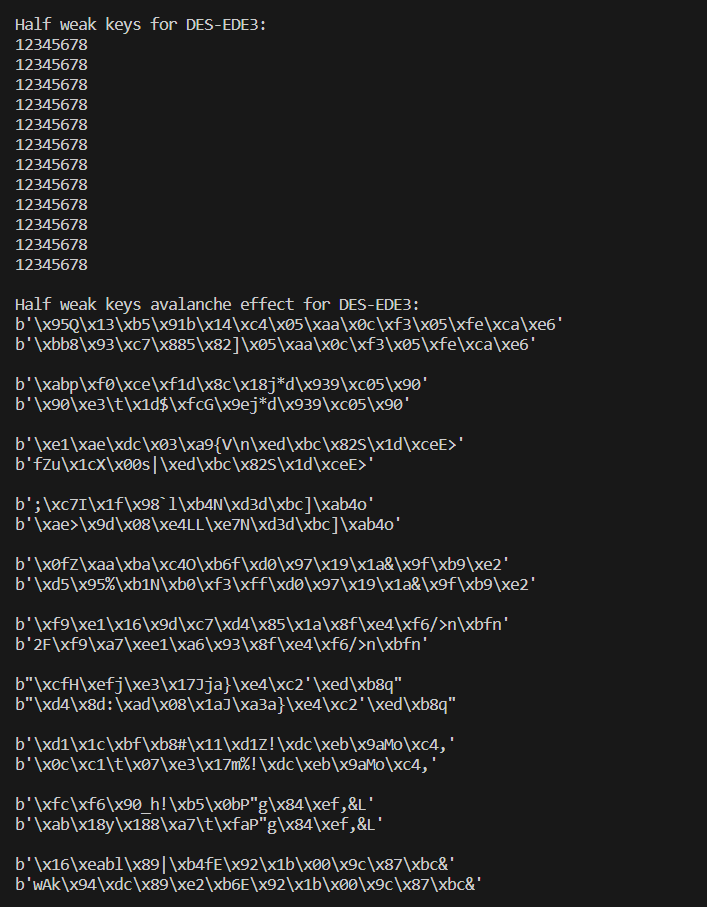


Рисунок 6 – Использование полуслабых ключей

Результаты анализа позволят определить, как слабые ключи и полуслабые ключи влияют на конечный результат зашифрования и на лавинный эффект. Это позволит определить, насколько важно использовать сильные ключи при шифровании данных и какие меры можно принять для улучшения безопасности данных при использовании блочных шифров.

3. Оценить степень сжатия (используя любой доступный архиватор) открытого текста и соответствующего зашифрованного текста. Дать пояснения к полученному результату.

Файлы с исходным сообщением, состоящим из 5000 символов, и зашифрованным сообщением были сжаты с помощью архиватора WinRAR. Размер сжатого файла с исходным сообщением стал меньше на 40%, в то время как архив с зашифрованным стал весить меньше на 46,67%.

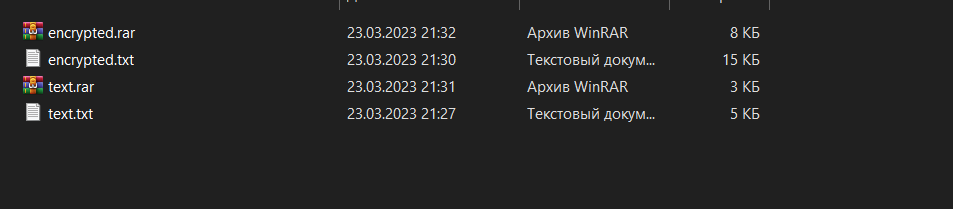


Рисунок 7 – Размеры исходных и заархивированных файлов

**Вывод:** блочные шифры широко применяются для защиты конфиденциальных данных, таких как пароли, кредитные карты, медицинские записи и другие.

Разработка и использование приложений для реализации блочных шифров требует глубоких знаний в области криптографии, математических основ криптографии и программирования. Для эффективной реализации и использования блочных шифров необходимо учитывать такие факторы, как скорость шифрования, сложность алгоритма, длина ключа и стойкость шифра.

**Контрольные вопросы**

1. Какие простейшие операции применяются в блочных алгоритмах шифрования?

Простейшие операции, которые применяются в блочных алгоритмах шифрования, включают в себя:

Побитовые операции: XOR, AND, OR, NOT - используются для комбинирования данных с ключом шифрования.

Перестановки - перемещение битов в блоке данных.

Замены - замена блоков данных определенными значениями с использованием таблиц замены (S-блоки).

Циклические сдвиги - смещение битов в блоке данных вправо или влево на фиксированное количество разрядов.

Обратимые линейные преобразования - преобразование блоков данных в линейное пространство и их изменение с помощью матриц.

2. В чем отличие блочных алгоритмов шифрования от потоковых?

Основное отличие между блочными и потоковыми алгоритмами шифрования заключается в том, как они обрабатывают данные.

Блочные алгоритмы шифрования обрабатывают данные блоками фиксированного размера (например, 64 или 128 бит), при этом каждый блок обрабатывается отдельно. Блочные алгоритмы работают с целыми блоками данных, которые могут быть большими или маленькими, в зависимости от конкретного алгоритма. Они могут использоваться для защиты как сообщений целиком, так и их частей.

Потоковые алгоритмы шифрования обрабатывают данные непрерывно, байт за байтом. Они не имеют фиксированного размера блока, а вместо этого шифруют поток битов. Потоковые алгоритмы работают с данными, поступающими на вход по мере их появления.

3. Что понимается под «раундом» алгоритма шифрования?

Раундом в криптографии понимается один шаг алгоритма шифрования, в котором происходит обработка данных для получения промежуточного результата.

4. Охарактеризовать и привести формальное описание сети Фейстеля.

Сеть Фейстеля – это тип блочных алгоритмов шифрования, который был предложен Хорстом Фейстелем в 1973 году. Она используется в таких известных алгоритмах, как DES, Blowfish, и Twofish.

Основная идея сети Фейстеля заключается в том, что исходный блок данных разбивается на две половины, и затем происходит ряд раундов, каждый из которых преобразует правую половину блока на основе левой половины и некоторого ключа. После всех раундов происходит обмен местами левой и правой половин блока и повторение процедуры до тех пор, пока не будет достигнуто нужное количество раундов или пока не будет достигнут конечный результат.

Формально, сеть Фейстеля может быть описана следующим образом:

Разделение блока данных на две половины: *L0* и *R0*.

Повторение *N* раундов:

a. Вычисление функции *f(Ri, Ki)*, где *Ri* - правая половина блока, *Ki* - ключ для *i*-го раунда, *f* - некоторая нелинейная функция, которая обычно является перестановкой битов.

b. Вычисление *Li+1 = Ri*

c. Вычисление *Ri+1 = Li XOR f(Ri, Ki)*

Обмен местами левой и правой половин блока: *Ln+1 = Rn* и *Rn+1 = Ln*

Ключи для каждого раунда могут быть получены из основного ключа шифрования с помощью алгоритма расширения ключа, который преобразует основной ключ в набор ключей, каждый из которых используется для шифрования соответствующего раунда.

Преимуществом сети Фейстеля является её простота, скорость работы и высокий уровень безопасности, который обеспечивается использованием нелинейной функции *f*.

6. В чем состоит особенность сложения чисел по модулю 2n?

Особенность сложения чисел по модулю 2^n заключается в том, что результатом операции сложения является остаток от деления суммы чисел на 2^n. То есть если мы складываем два числа a и b по модулю 2^n, то результат будет равен (a + b) mod 2^n.

Это означает, что при сложении двух чисел по модулю 2^n мы ограничиваем результат до n битов. Если сумма чисел превышает 2^n, то результатом будет остаток от деления этой суммы на 2^n. Таким образом, при сложении по модулю 2^n мы получаем результат в виде n-битного числа, которое можно хранить в регистре фиксированной длины.

7. Сложить по модулю 102 пары чисел: 55 и 14; 76 и 24; 99 и 99.

55 + 14 = 69. Результатом будет остаток от деления 69 на 102: 69 mod 102 = 69.

76 + 24 = 100. Результатом будет 100 mod 102 = 100.

99 + 99 = 198. Результатом будет 198 mod 102 = 96.

8. Сложить по модулю 28: двоичные числа 10101100 и 11001010; 01111111 и 01101101; шестнадцатеричные числа 0В5 и 37.

Для сложения двоичных чисел по модулю 2^8 мы должны использовать только последние 8 битов каждого числа и затем сложить эти биты по модулю 2^8.

1) В данном случае, последние 8 битов каждого числа равны:

10101100 -> 0101100

11001010 -> 1001010

Суммируя эти два числа, мы получаем:

0101100

1001010 =

1110110

Результат сложения по модулю 2^8 двоичных чисел 10101100 и 11001010 равен 1110110, или в десятичной системе счисления - 118.

2) В данном случае, последние 8 битов каждого числа равны:

01111111 -> 1111111

01101101 -> 1101101

Суммируя эти два числа, мы получаем:

1111111

1101101 =

11001000

Результат сложения по модулю $2^8$ двоичных чисел 01111111 и 01101101 равен 11001000, или в десятичной системе счисления - 200.

3) Переведем оба числа в двоичную систему счисления:

0В5 = 0000 1011 37 = 0011 0111

Теперь сложим их по модулю 2^8, то есть оставим только младшие 8 бит результата:

0000 1011

0011 0111=

0100 0010

Таким образом, сумма чисел 0В5 и 37 по модулю 2^8 равна 0100 0010, что в шестнадцатеричной системе счисления соответствует числу 42.

9. Дать пояснение принципам реализации «лавинного» эффекта.

"Лавинный" эффект в криптографии является явлением, при котором изменение даже одного бита входного сообщения или ключа приводит к значительному изменению выходного значения хеш-функции или шифра. Этот эффект является желательным свойством криптографических алгоритмов, так как он делает невозможным прогнозирование выхода алгоритма для незначительно измененных входных данных.

10. Выбрать два произвольных блочных алгоритма. В чем состоят отличия между ними?

Основное отличие между AES и DES заключается в их безопасности и скорости работы. AES является более безопасным и быстрым алгоритмом, по сравнению с DES. Он также использует ключи более длинной длины и имеет более эффективную структуру. DES является устаревшим алгоритмом, который стал подвержен атакам со стороны хакеров, и его использование уже не рекомендуется для защиты конфиденциальных данных.

11. Представить графически и пояснить функционал одного раунда блочного алгоритма DES (АES, ГОСТ 28147–89, Blowfish).

См. выше.

12. Сколько можно реализовать (теоретически) разновидностей алгоритма 3DES?

Количество возможных вариантов 3DES можно оценить, учитывая следующее:

* Существует три ключа, каждый из которых имеет длину 56 бит. Это означает, что можно создать 2^56 различных ключей для каждой из трех позиций.
* Существует два режима работы: EDE и EEE.
* Кроме того, можно изменить порядок ключей в режиме EDE, что дает еще один вариант.

Таким образом, общее число возможных вариантов 3DES будет следующим:

2^56 \* 2^56 \* 2^56 \* 2 \* 2 = 2^168 \* 4.

13. Какие факторы влияют на стойкость блочного алгоритма шифрования?

Стойкость блочного алгоритма шифрования зависит от нескольких факторов:

1. Длина ключа. Длина ключа должна быть достаточно большой, чтобы сделать перебор всех возможных ключей практически невозможным. Чем больше длина ключа, тем выше стойкость алгоритма.
2. Размер блока. Размер блока должен быть достаточно большим, чтобы затруднить вычисление статистических свойств шифротекста, основанных на анализе частоты появления отдельных символов или битов.
3. Количество раундов. Количество раундов должно быть достаточным, чтобы обеспечить хорошую диффузию и конфузию. Диффузия означает, что изменение одного бита в открытом тексте должно привести к изменению большого числа битов в шифротексте. Конфузия означает, что каждый бит шифротекста должен зависеть от максимального числа битов ключа и открытого текста.
4. Режим шифрования. Режим шифрования должен обеспечивать безопасность от возможных атак, связанных с дублированием блоков, а также от атак, связанных с известным открытым текстом или шифротекстом.
5. Криптографические методы. Важно использовать сильные криптографические методы, такие как подстановочно-перестановочные сети (SP-сети), хэш-функции и др., которые обеспечивают хорошую диффузию и конфузию.
6. Разнообразие ключей. Важно использовать разные ключи для разных блоков, чтобы уменьшить вероятность обнаружения структуры в шифротексте.

14. В чем состоит сущность дифференциального криптоанализа?

Сущность дифференциального криптоанализа заключается в анализе различий (дифференциалов) между парами открытого текста и соответствующих им шифротекстов, полученных с использованием одного и того же ключа.

Дифференциальный криптоанализ заключается в поиске таких пар открытого текста, для которых существует большая вероятность получения определенного различия в соответствующих им шифротекстах. Затем на основе этой информации атакующий может использовать статистические методы для вычисления ключа шифрования.

15. В чем состоит сущность линейного криптоанализа?

Сущность линейного криптоанализа заключается в анализе линейных зависимостей между битами ключа, открытым текстом и соответствующим шифротекстом. Линейный криптоанализ ищет линейную функцию (или комбинацию функций), которая наилучшим образом аппроксимирует зависимость между входными и выходными данными алгоритма шифрования.

16. Какие ключевые комбинации относятся к слабым (к полуслабым) и почему?

См. выше

17. Где применяются блочные криптоалгоритмы?

Блочные криптоалгоритмы широко применяются для защиты информации во многих областях, включая:

1. Компьютерная безопасность: блочные криптоалгоритмы используются для шифрования конфиденциальной информации в компьютерных сетях, интернет-передаче данных, защиты файлов, дисков и т.д.
2. Финансы: блочные криптоалгоритмы используются для защиты банковских транзакций и финансовых операций, включая электронные платежи, кредитные карты, банковские переводы и т.д.
3. Телекоммуникации: блочные криптоалгоритмы используются для защиты конфиденциальности телефонных звонков и сообщений, а также в сотовых сетях.
4. Государственная безопасность: блочные криптоалгоритмы широко используются для защиты государственной тайны и другой конфиденциальной информации.
5. Медицина: блочные криптоалгоритмы могут использоваться для защиты конфиденциальных медицинских данных пациентов, таких как медицинские записи, результаты анализов, истории болезни и т.д.
6. Промышленность: блочные криптоалгоритмы используются для защиты конфиденциальной промышленной информации, такой как патенты, проекты, технологии, изобретения и т.д.
7. Армия и военная безопасность: блочные криптоалгоритмы широко используются в военных целях для защиты конфиденциальных данных, таких как планы операций, геополитические данные, военные сообщения и т.д.
8. Личная безопасность: блочные криптоалгоритмы могут использоваться для защиты личных данных, таких как пароли, пин-коды, личная информация и т.д.

Таким образом, блочные криптоалгоритмы играют важную роль в обеспечении безопасности в различных сферах и областях.