Лабораторная работа № 5

Исследование блочных шифров

Цель: изучение и приобретение практических навыков разработки и использования приложений для реализации блочных шифров (рассчитана на 4 часа аудиторных занятий).

Задачи:

1. Закрепить теоретические знания по алгебраическому описанию, алгоритмам реализации операций зашифрования/расшифрования и оценке криптостойкости блочных шифров.
2. Разработать приложение для реализации указанных преподавателем методов блочного зашифрования/расшифрования.
3. Выполнить анализ криптостойкости блочных шифров.
4. Оценить скорость зашифрования/расшифрования реализованных шифров.
5. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

5.1 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

5.1.1 Краткая историческая информация и общая характеристика блочных шифров

В 1972 г. Национальное бюро стандартов США (ныне – Национальный институт стандартов и технологии, National Institute of Standarts & Technology – NIST) инициировал программу защиты каналов связи и компьютерных данных. Одна из целей – разработка единого стандарта криптографического шифрования. Основными критериями оценки алгоритма являлись [4]:

- алгоритм должен обеспечить высокий уровень защиты,
- алгоритм должен быть понятен и детально описан,
- криптостойкость алгоритма должна зависеть только от ключа,
- алгоритм должен допускать адаптацию к различным применениям,
- алгоритм должен быть разрешен для экспорта.

В качестве начального варианта нового алгоритма рассматривался Lucifer – разработка компании IBM начала семидесятых годов. В основе указанного алгоритма использовались два запатентованных 1971 г. Хорстом Фейстелем (Horst Feistel) устройства, реализующие различные алгоритмы шифрования,

позже получившие *шифр* (*сеть*) *Фейстеля* (Feistel cipher, Feistel network). В первой версии проекта Lucifer сеть Фейстеля не использовалась.

После многочисленных согласований, специальных конференций, где рассматривались, в основном вопросы криптостойкости алгоритма, подлежащего утверждению в качестве федерального стандарта, в ноябре 1976 г. был утвержден стандарт DES (Data Encryption Standard – стандарт шифрования данных). Предполагалось, что стандарт будет реализовываться только аппаратно.

В 1981 г. ANSI одобрил DES в качестве стандарта для публичного использования (стандарт ANSI X3.92), назвав его алгоритмом шифрования данных (Data Encryption Algorithm – DEA).

В 1987 году были разработаны алгоритмы FEAL и RC2. Сети Фейстеля получили широкое распространение в 1990-е годы — в годы появления таких алгоритмов, как *Blowfish* (1993), TEA (1994), RC5 (1994), CAST-128 (1996), XTEA (1997), XXTEA (1998), RC6 (1998) и других. На основе сети Фейстеля в 1990 году в СССР был принят в качестве ГОСТ 28147-89 стандарт шифрования.

Предполагалось, что DES будет сертифицироваться каждые 5 лет. Срок действия последнего сертификата на территории США истек практически к концу 20 века. К тому времени DES был вскрыт «лобовой атакой».

В 1998 г. NIST объявил конкурс на новый стандарт, который завершился в 2001 г. принятием AES (Advanced Encryption Standard).

Все перечисленные стандарты и алгоритмы *блочных шифров* (БШ) строятся на основе подстановочных и перестановочных, т. е. являются комбинационными. БШ относятся также к классу симметричных.

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*.

Основные требования к шифрам рассматриваемого класса можно сформулировать следующим образом:

- даже незначительное изменение исходного сообщения должно приводить к существенному изменению зашифрованного сообщения;
- устойчивость к атакам по выбранному тексту;
- алгоритмы зашифрования/расшифрования должны быть реализуемыми на различных платформах;
- алгоритмы должны базироваться на простых операциях;

• алгоритмы должны быть простыми для написания кода, вероятность появления программных ошибок должна быть низкой;

• алгоритмы должны допускать их модификацию при переходе на иные требования по уровню криптостойкости.

5.1.2 Сеть Фейстеля

Само название конструкции Фейстеля (сеть) означает ее *ячеистую* топологию [28]. Формально одна ячейка сети соответствует одному раунду зашифрования или расшифрования сообщения.

При зашифровании сообщение разбивается на блоки одинаковой (фиксированной) длины (как правило – 64 или 128 бит).

Полученные блоки называются *входными*. В случае, если длина входного блока меньше, чем выбранный размер, то блок удлиняется установленным способом.

Каждый входной блок шифруемого сообщения изначально делится на два подблока одинакового размера: левый (L_0) и правый (R_0). Далее в каждом i-ом раунде выполняются преобразования в соответствии с формальным представлением ячейки сети Фейстеля:

$$\begin{aligned} L_i &= R_{i-1}, \\ R_i &= L_{i-1} + f(R_{i-1},\ K_i), \end{aligned} \qquad (5.1)$$

По какому-либо математическому правилу вычисляется раундовый ключ K_i. В приведенном выражение знак «+» соответствует поразрядному суммированию на основе «XOR». На рис. 5.1 приведено графическое отображение сети Фейстеля.

Расшифрование происходит так же, как и зашифрование, с той лишь разницей, что раундовые ключи будут использоваться в обратном порядке по отношению к зашифрованию.

В своей статье [28] Х. Фейстель описывает два блока преобразований с использованием функции $f(R_{i-1},\ K_i)$:

блок подстановок (S-блок, англ. S-box);

блок перестановок (P-блок, англ. P-box).

Блок подстановок состоит из:

- дешифратора, преобразующего n-разрядное двоичное число в одноразрядное сигнал по основанию 2^n;
- внутреннего коммутатора;

- шифратора, преобразующего сигнала из одноразрядного 2^n-ричного в n-разрядный двоичный.
  Пример реализации трехразрядного S-блок показан на рис. 5.2.

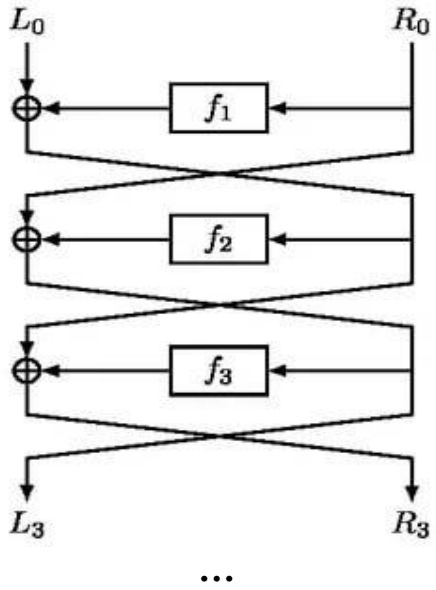

Рисунок 5.1 Графическое отображение сети Фейстеля

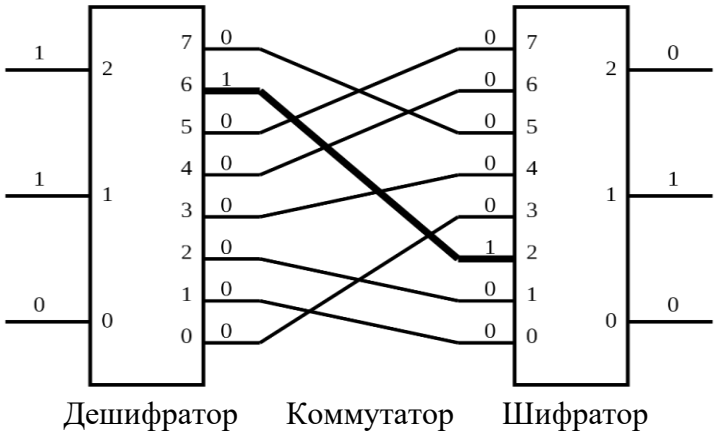

Рисунок 5.2 Пример реализации трехразрядного S-блока

Блок перестановок изменяет положение цифр, т. е. является линейным устройством. Этот блок может иметь очень большое количество входов-выходов, однако в силу линейность является слабым местом преобразования с

точки зрения криптостойкости. На рис. 5.3 приведен пример реализации 8-миразрядного P-блока.

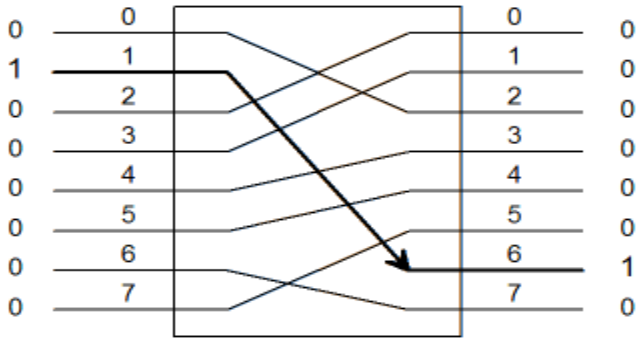

Рисунок 5.3 Пример реализации 8-миразрядного P-блока

Термин «блок» в оригинальной статье [28] используется вместо термина «функция» вследствие того, что речь идет о блочном шифре.

При большом размере блоков (128 бит и более) реализация сети Фейстеля на 32-разрядных архитектурах может вызвать затруднения, поэтому применяются модифицированные варианты сети. Такие модификации предусматривают использование не 2-х (*L* и *R* на рис. 5.1), а 4-х ветвей. На рис. 5.4 показаны некоторые модификации.

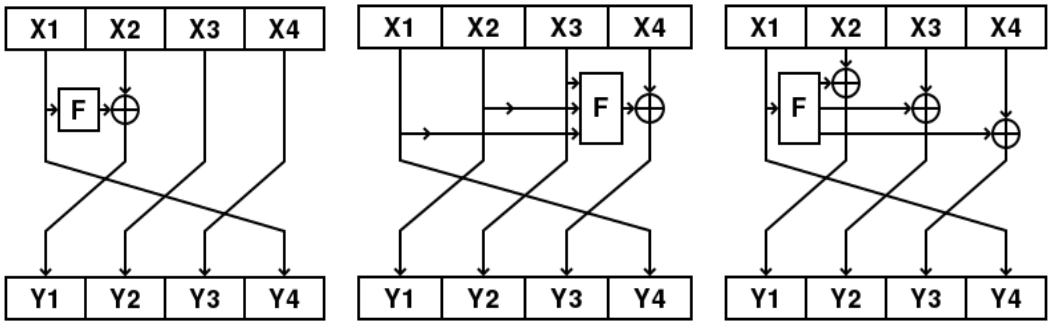

Рисунок 5.4 Примеры модификаций базовой сети Фейстеля

В основе *криптостойкости* блочных шифров лежит идея К. Шеннона в представлении составного шифра таким образом, чтобы от обладал двумя важными свойствами: *рассеянием* и *перемешиванием*. Рассеивание должно скрыть отношения между зашифрованным текстом и исходным текстом.

Рассеивание подразумевает, что каждый символ (символ или бит) в зашифрованном тексте зависит от одного или всех символов в исходном тексте.

Другими словами, если единственный символ в исходном тексте изменен, несколько или все символы в зашифрованном тексте будут также изменены.

Идея относительно перемешивания заключается в том, что оно должно скрыть отношения между зашифрованным текстом и ключом.

5.1.3 Базовые операции сложения чисел в блочных шифрах

Как было указано выше, в основе сети Фейстеля лежит простейшая операция суммирования 2-х ($A + B$) n-разрядных чисел – XOR: $A + B \pmod{n}$. Помимо этой операции некоторые алгоритмы (Blowfish, IDEA, ГОСТ и др.) предусматривают выполнение операций сложения чисел по модулю более высоких порядков: XOR: $A + B \pmod{2^n}$. Понятно, что числа A и B также являются n-разрядных.

Реализация второй из указанных операций является более сложной. Вспомним основные ее особенности.

Во-первых. Самое большое слагаемое меньше 2^n. Например, при n=3 самое большое слагаемое в двоичном виде – это 111 (или 7), а $2^n = 8$.

Во-вторых. Результатом сложения также должно быть n-разрядное число.

В-третьих. Побитовое сложение предусматривает известную взаимосвязь между соседними символами (порядками).

В-пятых. В силу известных правил модулярной1 арифметики результат вычисления $A + B \pmod{2^n}$ – это остаток от деления: $(A + B)/ 2^n$.

Рассмотрим примеры.

Пример 1. Пусть n=4, A = 9, B = 7, т. е. $2^n = 16$.

Легко подсчитать, $A + B$ = 16; 9 + 7 (mod 16) = 0.

Представим слагаемые в двоичной форме: A = 1001, B = 0111; $A + B$ =1001 + 0111 = 1**0000**. Для получения нужного результата – вычисления 9 + 7 (mod 16) – следует взять младшие 4 разряда (n=4) суммы: 0000.

Пример 2. Пусть n=4, A = 4, B = 10. Выполним вычисления по аналогии с примером 1.

Получаем $A + B$ = 14, результирующая сумма меньше 16; 14 (mod 16) = 14.

Представим слагаемые в двоичной форме: A = 0100, B = 1010; $A + B$ =0100 + 1010 = **1110**. Таким образом, итоговое число состоит из требуемых 4-х разрядов.

Пример 3. Пусть n=4, A = 10, B = 11. Их сумма по модулю 16 дает 5. В двоичной форме: 1010 + 1011 = 1**0101**. Искомое двоичное число – 4 младших разряда.

Если для данных в каждом из рассмотренных примеров мы проведем операцию деления $(A + B)/ 2^4$, получим те же 4 бита (выделены жирным).

Таким образом, общие правила выполнения рассматриваемой операции формально можно представить следующим образом:

$$A + B \pmod{2^n} = \left\{ \begin{array}{ll} A + B, & \text{если } A + B < 2^n \\ A + B - 2^n, & \text{если } A + B \geq 2^n \end{array} \right\}.$$

5.1.4 Некоторые блочные алгоритмы

5.1.3.1 Алгоритм DES

В силу причин, перечисленных в п.5.1.1, данный алгоритм является, пожалуй, наиболее исследованным.

Алгоритм строится на основе сети Фейстеля.

Входной блок данных, состоящий из 64 бит, 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.

Один блок данных подвергается преобразованию (и при зашифровании, и при расшифровании) в течение 16 раундов.

После первоначальной перестановки и разделения 64-битного блока данных на правую (R_0) и левую (L_0) половины длиной по 32 бита выполняются 16 раундов одинаковых действий (см. рис. 5.5.). Функционал этих действий подробно рассмотрен в п. 5.1 пособия [2].

В таблице 5.1 показан принцип первоначальной перестановки разрядов (*IP*) входного 64-битового слова.

Таблица 5.1

Начальная перестановка

58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
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

Выполненная перестановка означает, например, что первый бит входного блока сообщения будет размещен на 40-й позиции (цифра «1» выделена жирным), а 58-й (выделено жирным с подчеркиванием) – на первой и т.д. Из беглого анализа выполненной перестановки легко понять принцип. Алгоритм перестановки разрабатывался для облегчения загрузки блока входного сообщения в специализированную микросхему. Вместе с тем, эта операция придает некоторую "хаотичность" исходному сообщению, снижая возможность использования криптоанализа статистическими методами.

Левая и правая ветви каждого промежуточного значения обрабатываются как отдельные 32-битные значения, обозначенные L_i и R_i.

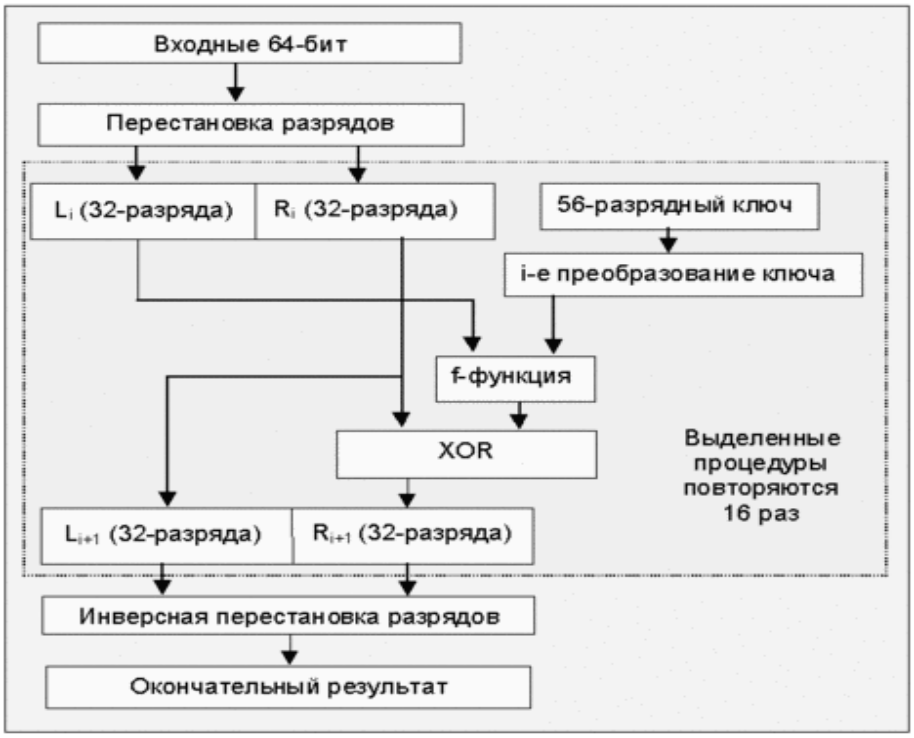

Рисунок 5.5 Общая схема алгоритма DES

Вначале правая часть блока R_i расширяется до 48 битов, используя таблицу, которая определяет перестановку плюс расширение на 16 битов. Эта операция приводит размер правой половины в соответствие с размером ключа для выполнения операции XOR. Кроме того, за счет выполнения этой операции быстрее возрастает зависимость всех битов результата от битов исходных данных и ключа (это называется «*лавинным эффектом*»).

После выполнения перестановки с расширением для полученного 48-битного значения выполняется операция XOR с 48-битным подключом K_i. Затем полученное 48-битное значение подается на вход блока подстановки *S* (от англ. *S*ubstitution – подстановка), результатом которой является 32-битное значение. Подстановка выполняется в восьми блоках подстановки или восьми *S*-блоках (*S*-boxes). При выполнении этой операции 48 битов данных делятся на восемь 6-битных подблоков, каждый из которых по соответствующей *таблице замен* замещается четырьмя битами. Подстановка с помощью S-блоков является одним из важнейших этапом 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 *P* (от англ. *P*ermutation – перестановка), которая не зависит от используемого ключа. Целью перестановки является максимальное переупорядочивание битов такое, 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-битового блока данных. Затем левая и правая половины меняются местами, и начинается следующий раунд (см. рис. 5.6).

После выполнения 16-раундового зашифрования 64-битного блока данных осуществляется конечная перестановка (IP^{-1}). Она является обратной к перестановке *IP*. Конечная перестановка определяется таблицей 5.2.

Таблица 5.2

Конечная перестановка

40	8	48	16	56	24	64	32	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

Рисунок 5.6. Схема реализации функции f на рис. 5.5

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 (см. лабораторные работы №№ 4-6 из [1]). 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 K_i. Ключ K_{16} используется в первом раунде, K_1 – в последнем.

После последнего раунда процесса расшифрования две половины выхода меняются местами так, чтобы вход заключительной перестановки был составлен из подблоков R_{16} и L_{16}. Выходом этой стадии является расшифрованный текст.

Слабые и полуслабые ключи. Из-за того, что первоначальный ключ изменяется при получении подключа для каждого раунда алгоритма, определенные

первоначальные ключи являются *слабыми* [4]. Вспомним, что первоначальное значение разделяется на две половины, каждая из которых сдвигается независимо. 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.

Четыре слабых ключа показаны в шестнадцатеричном виде в табл. 5.3 (каждый восьмой бит – это бит четности).

Таблица 5.3 [4]

Слабые ключи DES

0101	0101	0101	0101
1F1F	1F1F	0E0E	0E0E
E0E0	E0E0	F1F1	F1F1
FEFE	FEFE	FEFE	FEFE

Кроме того, некоторые пары ключей при зашифровании переводят открытый текст в идентичный шифртекст. Иными словами, один из ключей пары может расшифровать сообщения, зашифрованные другим ключом пары. Это происходит из-за метода, используемого DES для генерации подключей: вместо 16 различных подключей эти ключи генерируют только два различных подключа. В алгоритме каждый из этих подключей используется восемь раз. Эти ключи, называемые *полуслабыми*, в шестнадцатеричном виде приведены в табл. 5.4.

Таблица 5.4 [4]

Полуслабые ключи DES

01FE	01FE	01FE	01FE	и	FE01	FE01	FE01	FE01
1FE0	1FE0	0EF1	0EF1	и	E01F	E01F	F10E	F10E
01E0	01E0	01F1	01F1	и	E001	E001	F101	F101
1FFE	1EEE	0EFE	0EFE	и	FE1F	FE1F	FE0E	FE0E
011F	011F	010E	010E	и	1F01	1F01	0E01	0E01
E0FE	E0FE	F1FE	F1FE	и	FEE0	FEE0	FEE1	FEE1

Криптоанализ DES. *Дифференциальный криптоанализ* базируется на таблице неоднородных дифференциальных распределений *S*-блоков в блочном шифре. Криптоанализ шифртекстов на основе рассматриваемого стандарта «работает» с парами шифртекстов, открытые тексты которых имеют определенные разности, как это отмечалось в материалах к лабораторной работе № 2. Метод анализирует эволюцию этих разностей в процессе прохождения открытых текстов раундов DES при шифровании одним и тем же ключом.

Для DES термин «разность» определяется с помощью операции XOR. Затем, используя разности полученных шифртекстов, присваивают различные вероятности различным ключам. В процессе дальнейшего анализа следующих пар шифртекстов один из ключей станет наиболее вероятным. Это и есть правильный ключ (см. более подробно [4], с. 326).

Линейный криптоанализ. Для того, чтобы найти линейное приближение для DES нужно найти «хорошие» однораундовые линейные приближения и объединить их. Обратим внимание на *S*-блоки. У них 6 входных битов и 4 выходных. Входные биты можно объединить с помощью операции XOR 63 способами (2^6 - 1), а выходные биты – 15 способами. Теперь для каждого *S*-блока можно оценить вероятность того, что для случайно выбранного входа входная комбинация XOR равна некоторой выходной комбинации XOR. И т.д.

DES давно характеризуется низкой криптостойкостью: в январе 1999 г. закодированное посредством DES сообщение было взломано с помощью связанных через Internet в единую сеть 100 тыс. персональных компьютеров за 24 часа. Данному алгоритму присуща проблема так называемых «слабых» и «частично слабых» ключей [4].

Основное достоинство DES – относительно высокая скорость (из-за малой длины ключа); бесплатное распространение по всему миру; общедоступность и отсутствие необходимости лицензионных отчислений.

Модификацией DES является 3DES. Создан У. Диффи, М. Хеллманом, У. Тачманном в 1978 г.

Формальная запись:

$$C_i = f(M_j,(\mathrm{DES}(K_3, (\mathrm{DES}(K_2, (\mathrm{DES}, K_1)))))).$$

Существуют несколько реализаций алгоритма 3DES. Вот некоторые из них:

- DES-EEE3: шифруется 3 раза с 3 разными ключами (операции шифрование-шифрование-шифрование);
- DES-EDE3: 3DES операции шифрование-расшифрование-шифрование с разными ключами:
- DES-EEE2 и DES-EDE2: как и предыдущие, однако, на первом и третьем шаге используется одинаковый ключ.

Расшифрование происходит, как и в простом DES, в обратном порядке по отношению к процедуре зашифрования.

3DES с тремя ключами реализован во многих Интернет-приложениях. Например, в PGP (Pretty Good Privacy) – позволяет выполнять операции шифрования и цифровой подписи сообщений, файлов и другой информации, представленной в электронном виде, например, на жёстком диске); в S/mime для обеспечения криптографической безопасности электронной почты.

3DES используется при управлении ключами в стандартах ANSI X9.17 (метод генерации 64-битных ключей) и ISO 8732 (управление ключами в банковском деле), а также в PEM (Privacy Enhanced Mail).

5.1.3.2 Стандарт AES

AES (Advanced Encryption Standard) – алгоритм шифрования, действующий в качестве государственного стандарта в США с 2001 года. В основу стандарта положен шифр *Rijndael*. Шифр *Rijndael/AES* (то есть рекомендуемый стандартом) характеризуется размером блока 128 бит, длиной ключа 128, 192 или 256 бит и количеством раундов 10, 12 или 14 в зависимости от длины ключа.

Основу *Rijndael* составляют так называемые линейно-подстановочные преобразования. В алгоритме широко используются табличные вычисления, причем все необходимые таблицы задаются константно, т. е. не зависят ни от ключа, ни от данных.

5.1.3.3 Стандарт ГОСТ 28147-89

В Советском Союзе, в том числе – в Беларуси и в России в качестве стандарта на блочные алгоритмы шифрования с закрытым ключом в 1989 г. был принят ГОСТ 28147-89 [30].

ГОСТ предусматривает три режима шифрования (*простая замена*, *гаммирование*, *гаммирование с обратной связью*) и один режим выработки *имитовставки*. Первый из режимов шифрования предназначен для шифрования ключевой информации и не может использоваться для шифрования других данных, для этого предусмотрены два других режима. Режим выработки имитовставки (криптографической контрольной комбинации) предназначен для имитозащиты шифруемых данных, т.е. для их защиты от случайных или преднамеренных несанкционированных изменений.

Шифр ГОСТ 28147-89 построен по тем же принципам, что и американский DES, однако по сравнению с DES первый более удобен для программной реализации. В ГОСТ 28147-89 применяется более длинный ключ – 256 бит, здесь используются 32 раунда шифрования

Таким образом, основные параметры алгоритма криптографического преобразования данных ГОСТ 28147-89: размер блока составляет 64 бита, размер ключа – 256 бит, количество раундов – 32.

Алгоритм представляет собой классическую сеть Фейстеля. Шифруемый блок данных разбивается на две одинаковые части, правую *R* и левую *L*. Правая часть складывается по модулю 2^{32} с подключом раунда и посредством принятого алгоритма шифрует левую часть. Перед следующим раундом левая и правая части меняются местами. Такая структура позволяет использовать один и тот же алгоритм как для зашифрования, так и для расшифрования блока (рис. 5.7).

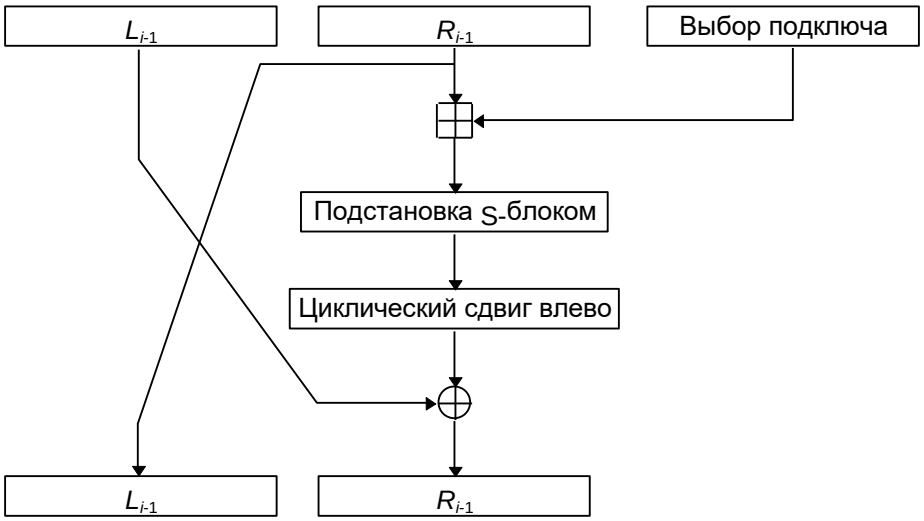

Рисунок 5.7 Структура алгоритма реализации раунда в стандарте ГОСТ 28147-89 (знаком «+» в квадратной рамке обозначена операция сложения по модулю 2^{32})

Таким образом, в алгоритме используются следующие операции:

- сложение слов по модулю 2^{32}: правый блок (R_i) складывается по модулю 2^{32} с текущим подключом (K_i);
- циклический сдвиг слова влево на указанное число бит;
- побитовое сложение по модулю 2 (XOR);
- замена (подстановка в блоке S) по таблице.

На различных шагах алгоритмов ГОСТа данные, которыми они оперируют, интерпретируются и используются различным образом. В некоторых случаях элементы данных обрабатываются как массивы независимых битов, в других случаях – как целое число без знака, в третьих – как имеющий структуру сложный элемент, состоящий из нескольких более простых элементов.

Сначала правый блок складывается по модулю 2^{32} с *подключом*. Полученное 32-битное сообщение делится на восемь 4-битных чисел. Каждое из этих 4-битных чисел преобразуется соответствующим S-блоком в другое 4-битное

число. Поэтому любой S-блок определяется некоторой 16-битной перестановкой на множестве из 16 элементов 0, 1, … , 15.

Исходный 256-битный ключ делится на восемь 32-битных подключей. Они используются в 32 тактах в следующем порядке: 1, 2, 3, 4, 5, 6, 7, 8, 1, 2, 3, 4, 5, 6, 7, 8, 1, 2, 3, 4, 5, 6, 7, 8, 8, 7, 6, 5, 4, 3, 2, 1. При расшифровании порядок использования подключей меняется на противоположный.

Поскольку в ГОСТ 28147-89 используется 256-битовый ключ, то объем ключевого пространства составляет 2^{256}. Даже, если предположить, что на взлом шифра брошены все силы вычислительного комплекса с возможностью перебора 10^{12} (это примерно равно 2^{40}) ключей в 1 секунду, то на полный перебор всех 2^{256} ключей потребуется 2^{216} секунд. Это время составляет более миллиарда лет.

К уже отмеченным отличиям между алгоритмами DES и ГОСТ можно добавить также следующее. В основном раунде DES применяются нерегулярные перестановки исходного сообщения, в ГОСТ используется 11-битный циклический сдвиг влево. Последняя операция гораздо удобнее для программной реализации. Однако перестановка DES увеличивает лавинный эффект. В ГОСТ изменение одного входного бита влияет на один 4-битовый блок при замене в одном раунде, который затем влияет на два 4-битовых блока следующего раунда, три блока, следующего и т. д. В ГОСТ требуется 8 раундов прежде, чем изменение одного входного бита повлияет на каждый бит результата; DES для этого нужно только 5 раундов.

5.1.3.4 Алгоритм Blowfish

Основой алгоритма является сеть Фейстеля с 16 раундами. Длина блока равна 64 битам, ключ может иметь любую длину в пределах 448 бит. Хотя перед началом любого зашифрования выполняется сложная фаза инициализации, само зашифрование данных выполняется достаточно быстро.

Алгоритм предназначен в основном для приложений, в которых ключ меняется нечасто, к тому же существует фаза начального рукопожатия, во время которой происходит аутентификация сторон. Blowfish характеризуется более высокой скоростью обработки боков, чем DES.

Алгоритм состоит из двух частей: расширение ключа и зашифрование/расшифрование данных. Расширение ключа преобразует ключ длиной, по крайней мере, 448 бит в несколько массивов подключей общей длиной 4168 байт.

Каждый раунд состоит из перестановки, зависящей от ключа, и подстановки, зависящей от ключа и данных. Операциями являются XOR и сложение по модулю 2^{32} (см. рис.

Blowfish использует большое количество подключей. Эти ключи должны быть вычислены заранее, до начала любого зашифрования/расшифрования данных.

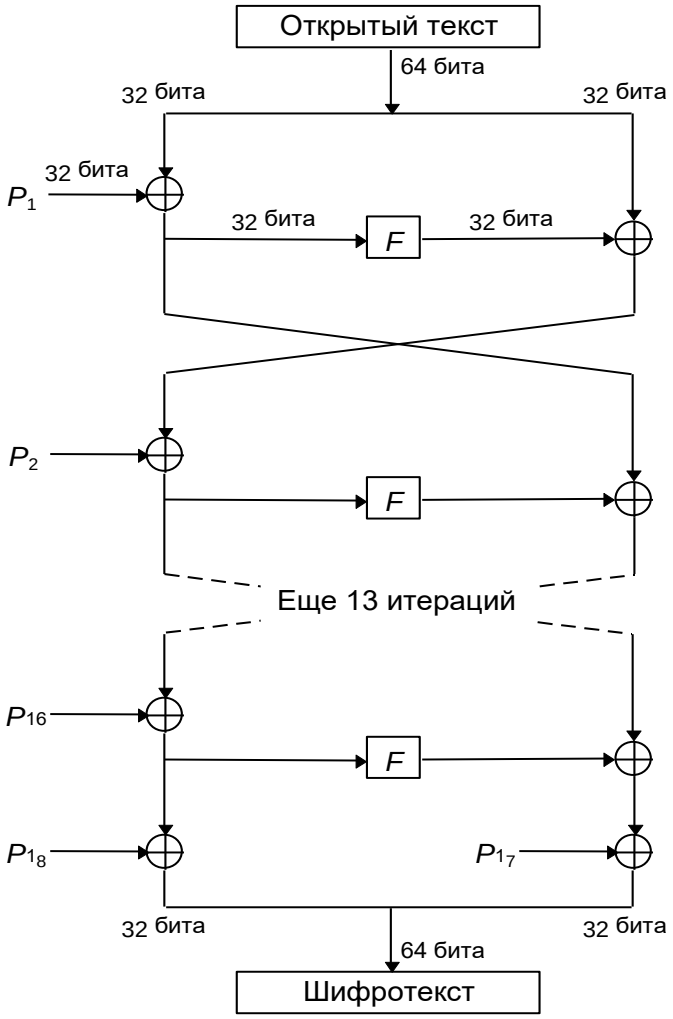

Рисунок 5.8 Структура алгоритма Blowfish

Элементы алгоритма:

- *P*-массив, состоящий из восемнадцати 32-битных подключей: P_1, P_2, ..., P_{18};
- *S*-блоки: каждый из четырех 32-битных *S*-блоков содержит 256 элементов;
- функция *F*, структура которой показана на рис. 5.9.

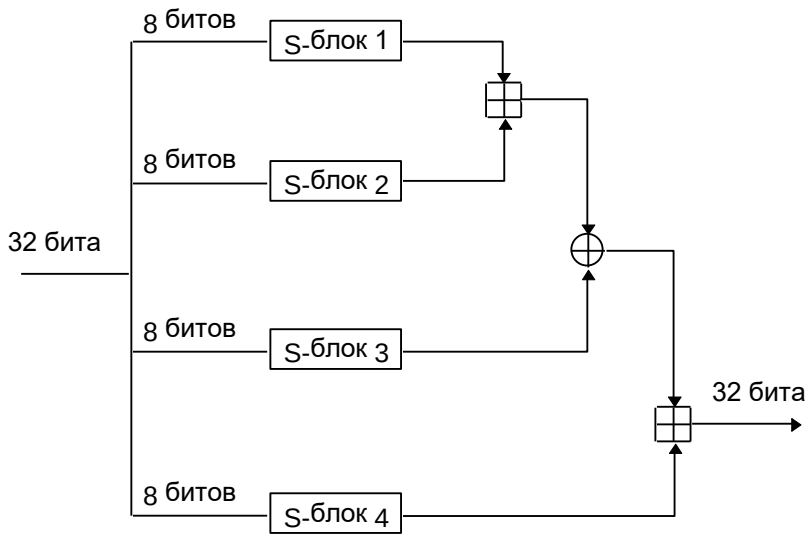

Рисунок 5.9 Структура функции *F* алгоритма Blowfish

Успешные атаки на рассмотренный алгоритм не известны.

Дополнительные сведения по рассмотренным вопросам можно найти в [8].

Следует также обратить внимание на еще одно обстоятельство: файл, который содержит зашифрованные данные, практически нельзя сжать. Это может служить одним из критериев поиска зашифрованных файлов, которые, как и обычные файлы, представляют собой набор случайных битов.

5.2 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, из приложения в [4].

Приложение должно реализовывать следующие операции:

- разделение входного потока данных на блоки требуемой длины с необходимым дополнением последнего блока;
- выполнение требуемых преобразований ключевой информации;
- выполнение операций зашифрования/расшифрования;
- оценка скорости выполнения операций зашифрования/расшифрования;
- пошаговый анализ лавинного эффекта с подсчетом количества изменяющихся символов по отношению к исходному слову.

Исследуемый метод шифрования и ключевая информация – в соответствии с вариантом из нижеследующей табл. 5.5.

Таблица 5.5

Вариант задания	Алгоритм	Ключ
1	DES	Первые 8 символов собственных *фамилииимени*
2	DES-EEE3	а) Информац*ионнаябе*зопаснос* б) лаборато*рная--ра*бота__№5* *каждый из трех ключей выделен шрифтом
3	DES-EDE3	По указанию преподавателя
4	DES-EEE2	По указанию преподавателя
5	DES-EDE2	По указанию преподавателя

По желанию студент может разработать приложение и выполнить связанные исследования для любого другого блочного алгоритма, не указанного в табл. 5.5.

2. Проанализировать влияние слабых ключей (табл. 5.3) и полуслабых ключей (табл. 5.4) на конечный результат зашифрования и на лавинный эффект.

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

4. Результаты оформить в виде отчета по установленным правилам.

ВОПРОСЫ ДЛЯ КОНТРОЛЯ И САМОКОНТРОЛЯ

1. Какие простейшие операции применяются в блочных алгоритмах шифрования?

2. В чем отличие блочных алгоритмов шифрования от потоковых?

3. Что понимается под "раундом" алгоритма шифрования?

4. Охарактеризовать и привести формальное описание сети Фейстеля.

5. Какие стандартные операции используются в блочных алгоритмах шифрования?

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

7. Сложить по модулю 10^2 пары чисел:
   55 и 14; 76 и 24; 99 и 99.

8. Сложить по модулю 2^8:
   двоичные числа 10101100 и 11001010; 01111111 и 01101101;
   шестнадцатеричные числа 0B5 и 37.

9. Дать пояснение принципам реализации «лавинного» эффекта.

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

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

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

13. Какие факторы влияют на стойкость блочного алгоритма шифрования?

14. В чем состоит сущность дифференциального криптоанализа?

15. В чем состоит сущность линейного криптоанализа?

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

17. Где применяются блочные криптоалгоритмы?