ЛАБОРАТОРНАЯ РАБОТА № 11

СЕТЬ ФЕЙСТЕЛЯ

Цель работы: изучение принципа работы сети Фейстеля.

Описание лабораторной работы. В 1971 году Хорст Фейстель (*Horst Feistel*) запатентовал два устройства для реализации различных алгоритмов шифрования, получившие название «Люцифер» (*Lucifer*). Одно из устройств использовало конструкцию, впоследствии названную «сетью Фейстеля» (*Feistel cipher*, *Feistel network*). Работа над созданием новых криптосистем велась Фейстелем в стенах IBM вместе с Доном Копперсмитом (*Don Coppersmith*). Проект «Люцифер» был скорее экспериментальным, но стал базисом для алгоритма *DES* (*Data Encryption Standard*). В 1973 году Хорст Фейстель в журнале *Scientific American* опубликовал статью «Криптография и Компьютерная безопасность» (*Cryptography and Computer Privacy*), в которой раскрыл ряд важных аспектов шифрования и привел описание первой версии проекта «Люцифер».

Описание алгоритма. Сеть Фейстеля подразумевает разбиение обрабатываемого блока данных на несколько субблоков (чаще всего — на два), один из которых обрабатывается некоей функцией $f(\alpha)$ и накладывается на один или несколько остальных субблоков. На рисунке 3.4 приведена наиболее часто встречающаяся структура алгоритмов на основе сети Фейстеля.

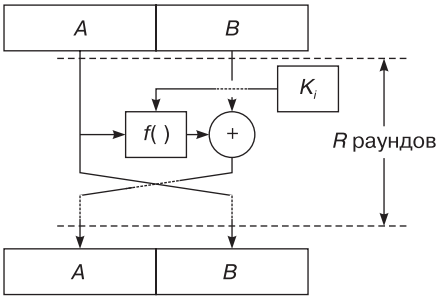

Рис. 3.4. Сеть Фейстеля

Блок открытого текста делится на две равные части A — левая (L) и B — правая (R), в каждом раунде i ($i = 1 \ldots n$ — номер раунда) вычисляется

$$L_i = R_{i-1} \oplus f(L_{i-1}, K_{i-1});$$
$$R_{i-1} = L_{i-1},$$

где $f()$ — некоторая функция, а K_{i-1} — ключ i-го раунда.

Результатом выполнения n раундов является (L_n, R_n), но обычно в n-ом раунде перестановка L_n и R_n не производится, что позволяет использовать ту же процедуру и для расшифрования, просто инвертировав порядок использования раундовой ключевой информации:

$$L_{i-1} = R_i \oplus f(L_i, K_{i-1});$$
$$R_{i-1} = L_i.$$

Одно из преимуществ такой модели — обратимость алгоритма независимо от используемой функции f, причем она может быть сколь угодно сложной.

Надежность алгоритма. Сети Фейстеля были широко изучены криптографами в силу их обширного распространения. В 1988 году Майкл Люби (*Michael Luby*) и Чарльз Ракофф (*Charles Rackoff*) провели исследования сети Фейстеля и доказали, что если раундовая функция является криптостойкой псевдослучайной и используемые ключи независимы в каждом раунде, то трех раундов будет достаточно для того, чтобы блочный шифр являлся псевдослучайной перестановкой.

Иногда сеть Фейстеля в западной литературе называют *Luby — Rackoff block cipher* в честь Люби и Ракоффа, которые проделали большой объем теоретических исследований в этой области.

В дальнейшем, в 1997 г., Мони Наор (*Moni Naor*) и Омер Рейнголд (*Omer Reingold*) предложили упрощенный вариант конструкции Люби — Ракоффа из четырех раундов, где в качестве первого и последнего раунда используются две попарно независимые перестановки. Два средних раунда конструкции Наора — Рейнголда идентичны раундам в конструкции Люби — Ракоффа.

Во многих блочных шифрах на основе сети Фейстеля были найдены те или иные уязвимости, однако в ряде случаев эти уязвимости являются чисто теоретическими и при нынешней производительности компьютеров использовать их на практике для взлома невозможно.

Симметричные криптоалгоритмы, использующие сеть Фейстеля. Так как большинство блочных шифров создано на основе сетей Фейстеля, коротко упомянем некоторые из них.

DES

DES работает с 64-битовым блоком открытого текста. После первоначальной перестановки блок разбивается на правую и левую половины длиной по 32 бита. Затем выполняются 16 этапов одинаковых действий, определяемых функцией f, в которых данные объединяются с ключом. После шестнадцатого этапа правая и левая половины объединяются и алгоритм завершается заключительной перестановкой (обратной по отношению к первоначальной).

На каждом этапе биты ключа сдвигаются и затем из 56 битов ключа выбираются 48 битов для цикловых ключей. Правая половина данных увеличивается до 48 битов с помощью перестановки с расширением, объединяется посредством XOR с 48 битами смещенного и переставленного ключа, проходит через восемь *S*-блоков, образуя 32 новых бита, и переставляется снова. Эти четыре операции и выполняются функцией *f*. Затем результат функции *f* объединяется с левой половиной с помощью другого XOR. В итоге этих действий появляется новая правая половина, а старая правая половина становится новой левой. Эти действия повторяются 16 раз, образуя 16 циклов.

FEAL

В качестве входа процесса шифрования используется 64-битовый блок открытого текста. Сначала блок данных подвергается операции XOR с 64 битами ключа. Затем блок данных расщепляется на левую и правую половины. Объединение левой и правой половин с помощью XOR образует новую правую половину. Левая половина и новая правая половина проходят через *n* этапов (первоначально четыре). На каждом этапе правая половина объединяется с помощью функции *f* с шестнадцатью битами ключа, и с помощью XOR — с левой половиной, создавая новую правую половину. Исходная правая половина (на начало этапа) становится новой левой половиной. После *n* этапов (левая и правая половины не переставляются после *n*-го этапа) левая половина снова объединяется с помощью XOR с правой половиной, образуя новую правую половину, затем левая и правая соединяются вместе в 64-битовое целое. Блок данных объединяется с помощью XOR с другими 64 битами ключа, и алгоритм завершается.

Функция *f* использует 32 бита данных и 16 битов ключа и смешивает их вместе. Сначала блок данных разбивается на 8-битовые подблоки, которые затем объединяются с помощью XOR и меняются местами.

RC2

RC2 — это шифр с 64-битовым блоком и переменной длиной ключа, разработанный как альтернатива DES. В соответствии с утверждениями компании *RSA Data Security* программные реализации RC2 в три раза быстрее DES. Алгоритм может использовать ключ переменной длины, от 0 байтов до максимальной длины строки, поддерживаемой компьютерной системой, причем скорость шифрования не зависит от размера ключа. Этот ключ предварительно используется для заполнения 128-байтовой таблицы, зависящей от ключа. RC2 не использует *S*-блоков, здесь используются две операции — «смешивание» и «перемешивание» (*mix* и *mash*), для каждого этапа выбирается одна из них.

RC2 не является итеративным блочным шифром. Это предполагает, что RC2 более устойчив к дифференциальному и линейному криптоанализу, чем другие блочные шифры, безопасность которых опирается на копировании схемы DES.

ГОСТ-28147—89

ГОСТ является 64-битовым алгоритмом с 256-битовым ключом. ГОСТ также использует дополнительный ключ в виде восьми различных *S*-блоков, функционирование которых рассматривается ниже. В процессе шифрования на 32 этапах последовательно выполняется алгоритм шифрования Фейстеля.

Для шифрования текст сначала разбивается на левую половину L и правую половину R. На этапе i алгоритма ГОСТ используется подключ K_i и выполняются следующие действия:

$$L_i = R_{i-1};$$
$$R_i = L_{i-1} f(R_{i-1}, K_i).$$

Функция f проста. Сначала правая половина и i-й подключ складываются по модулю 2^{32}. Результат разбивается на восемь четырехбитовых подблоков, каждый из которых поступает на вход своего *S*-блока. ГОСТ использует восемь различных *S*-блоков, первые четыре бита попадают в первый *S*-блок, вторые четыре бита — во второй *S*-блок, и т.д. Каждый *S*-блок представляет собой перестановку чисел от 0 до 15.

В этом случае, если на входе *S*-блока 0, то на выходе 7. Если на входе 1, на выходе 10, и т.д. Все восемь *S*-блоков в ГОСТ-28147—89 различны, они фактически являются дополнительным ключевым материалом. *S*-блоки должны храниться в секрете.

Выходы всех восьми *S*-блоков объединяются в 32-битовое слово, затем все слово циклически сдвигается влево на 11 битов. Наконец результат объединяется с помощью XOR с левой половиной и получается новая правая половина, а правая половина становится новой левой половиной. Количество таких циклов — 32.

Генерация подключей проста. 256-битовый ключ разбивается на восемь 32-битовых блоков: $k1$, $k2$, ... $k8$. На каждом этапе используется свой подключ. Расшифование выполняется также как и шифрование, но инвертируется порядок подключей k_i.

ГОСТ-28147—89 не определяет способ генерации *S*-блоков, говорится только, что блоки должны быть предоставлены каким-то образом. Это породило домыслы о том, что советский производитель может поставлять хорошие *S*-блоки «хорошим» организациям и плохие *S*-блоки тем организациям, которых производитель собирается надуть,

вследствие чего неофициальные переговоры с российским производителем микросхем ГОСТ выявили другую альтернативу. Производитель создает перестановки *S*-блока самостоятельно с помощью генератора случайных чисел.

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СЕТИ ФЕЙСТЕЛЯ

```
/* функция преобразования подблока по ключу (зависит
от конкретного алгоритма)
subblock — преобразуемый подблок
key — ключ
возвращаемое значение — преобразованный блок*/
int f (int subblock, int key);
/* Шифрование открытого текста
left — левый входной подблок
right — правый входной подблок
* key — массив ключей (по ключу на раунд)
rounds — количество раундов*/
void crypt (int *left, int *right, int rounds, int *key)
{
     int i, temp;
     for (i = 0; i < rounds; i++)
     {
        temp = *right ^ f (*left, key[i]);
        *right = *left;
        *left = temp;
     }
}
/* Расшифрование текста
left — левый зашифрованный подблок
right — правый зашифрованный подблок*/
void decrypt (int *left, int *right, int rounds, int *key)
{
     int i, temp;
     for (i = rounds — 1; i > = 0; i--)
     {
        temp = *left ^ f (*right, key [i]);
        *left = *right;
        *right = temp;
     }
}
```

Описание программного интерфейса. Главное окно программы представлено на рис. 3.5.

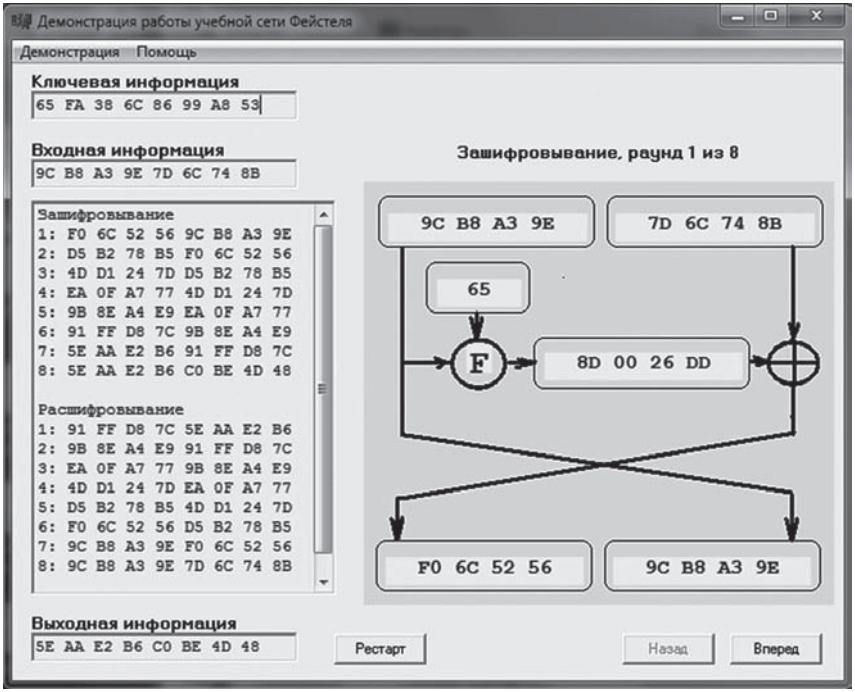

Рис. 3.5. Главное окно программы

Демонстрация

Рестарт — генерация случайного ключа (64 бит) и входной информации (64 бит).

Шаг вперед — переход к следующему раунду.

Шаг назад — переход к предыдущему раунду.

Выход — выход из программы.

Для удобства работы с программой основные функции продублированы кнопками в основном окне программы.

Помощь

Справка — содержит теоретическую информацию об алгоритме шифрования сеть Фейстеля и об использовании программы (рис. 3.6). Информация о разработчике представлена на рис. 3.7.

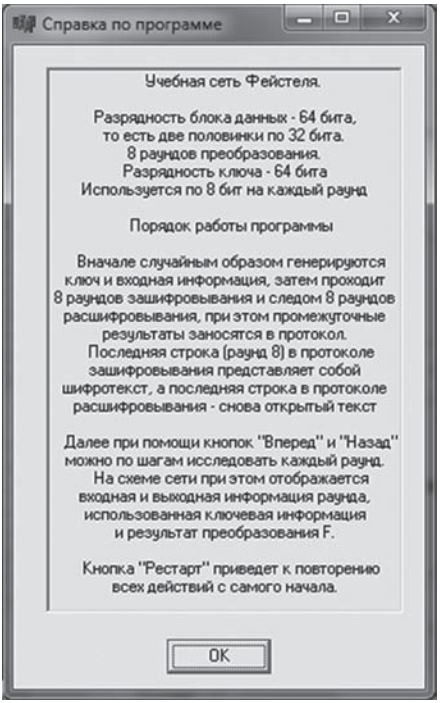

Рис. 3.6. Справочная информация

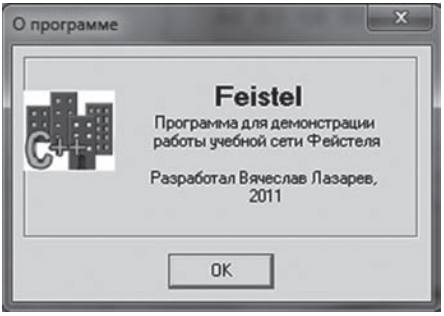

Рис. 3.7. Информация о разработчике

Поле **«Ключевая информация»** содержит случайно сгенерированный ключ.

Поле **«Входная информация»** содержит случайно сгенерированную информацию длиной 64 бита.

Поле **«Выходная информация»** содержит зашифрованную информацию.

Поле **«Зашифровывание»** отображает процесс шифрования, состоящий из восьми последовательных раундов.

Порядок выполнения программы

Нажать кнопку **«Рестарт»**, далее, используя клавиши вперед (назад), ознакомиться с процессом шифрования входной информации.

Задание

Для выполнения лабораторной работы на компьютере необходимо установить программу *Feistei. exe*. Запустить программу *Feistei. exe*, используемую для демонстрации работы сети Фейстеля.

1. Чтобы начать работу с программой, необходимо в Меню **«Демонстрация»** открыть вкладку **«Рестарт»** или нажать кнопку **«Рестарт»**.

2. С помощью переключателя **«Шаг вперед»** или кнопки **«Вперед»** можно последовательно наблюдать, каким образом шифруется на каждом цикле преобразования сети Фейстеля блок исходного текста.

3. Сохранить в отчете экранные формы, демонстрирующие процесс пошаговой работы сети Фейстеля при шифровании исходного текста.

4. Произвести вручную процесс шифрования в одном цикле и убедиться в том, что результаты, демонстрируемые программой, совпадают с произведенными расчетами. Сделать выводы по проделанной работе.

5. Включить в отчет о лабораторной работе ответы на контрольные вопросы, выбранные в соответствии с номером варианта из табл. 3.3.

Таблица 3.3

Номер варианта	Контрольные вопросы
1, 5, 7, 3, 9, 18, 28	В каких современных симметричных системах шифрования используется сеть Фейстеля?
2, 4, 6, 8, 20, 22, 24, 26, 30	В чем отличие сбалансированной сети Фейстеля от несбалансированной сети Фейстеля? В каких блочных криптосистемах используется сбалансированная сеть?
11, 13, 15, 10, 17, 19, 27	В каких современных симметричных системах шифрования не используется сеть Фейстеля? Какие механизмы шифрования используются в этих криптографических системах?
12, 14, 16, 21, 23, 25, 29	Какой длины используются блоки для шифрования и цикловые ключи в блочных криптосистемах DES, FIAL, Blowfish, ГОСТ-28147—89?