1. Цель лабораторной работы:

Реализовать алгоритм шифрования Blowfish. Предусмотреть возможность работы алгоритма в режиме OFB.

2. Описание шифра:

BlowFish — алгоритм 64-битного блочного шифра с ключом переменной длины. Был разработан известным специалистом в области криптографии и защиты информации Брюсом Шнайером в 1993 году.

Отличительными особенностями этого алгоритма стала более высокая степень криптостойкости, нежели алгоритма DES (в том числе за счет использования переменной длины ключа, до 448 бит) и высокая скорость шифрации/дешифрации (за счет генерации таблиц замены).

В общем случае алгоритм состоит из двух этапов — расширение ключа и шифрация/дешифрация исходных данных.

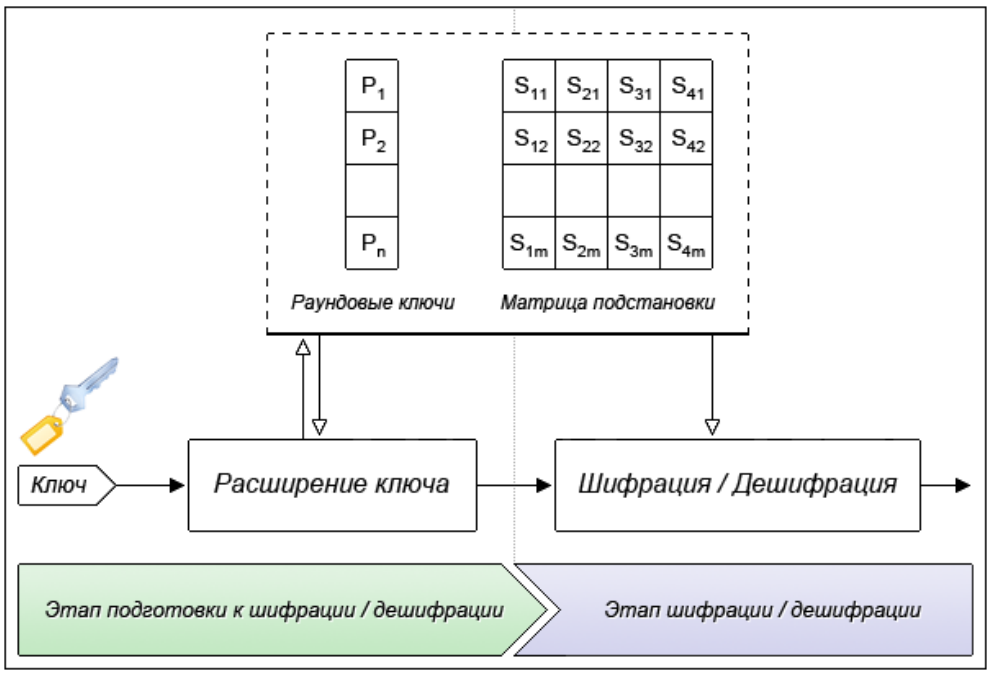


Рис. 1 Общая схема алгоритма Blowfish

На этапе расширения ключа, исходный ключ преобразуется в матрицу раундовых ключей и матрицу подстановки (или замены), общим объемом в 4168 байт.

Шифрация данных, а также создания матрицы раундовых ключей и подстановки, происходит через использование сети Фейстеля, состоящей в свою очередь из 16 раундов.

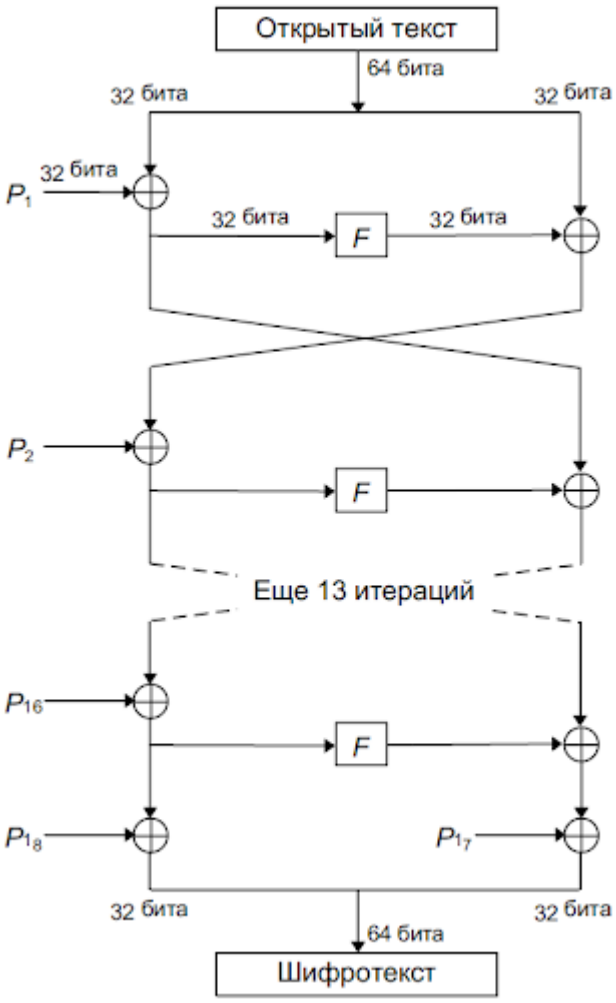


Рис. 2 Сеть Фейстеля для алгоритма Blowfish

Принцип работы сети Фейстеля для алгоритма Blowfish:

1. Исходные данные разбиваются на блоки фиксированной длины (как правило кратно степени двойки — 64 бит, 128 бит). В случае если длина блока исходных данных меньше длины разрядности шифра, то блок дополняется каким-либо заранее известным образом.

2. Блок делится на два равных подблока — «левый» L0 и «правый» R0. В случае 64-битной разрядности — на два блока с длиной 32 бита каждый.

3. «Левый подблок» L0 видоизменяется функцией итерации F(L0, P0) в зависимости от ключа P0, после чего он складывается по модулю 2 (XOR) с «правым подблоком» R0.

4. Результат сложения присваивается новому левому подблоку L1, который становится левой половиной входных данных для следующего раунда, а «левый подблок» L0 присваивается без

изменений новому правому подблоку R1, который становится правой половиной.

5. Эта операция повторяется n-1 раз, при этом при переходе от одного этапа к другому меняются раундовые ключи (P_0, P_1, P_2 и т.д.), где n — количество раундов для используемого алгоритма.

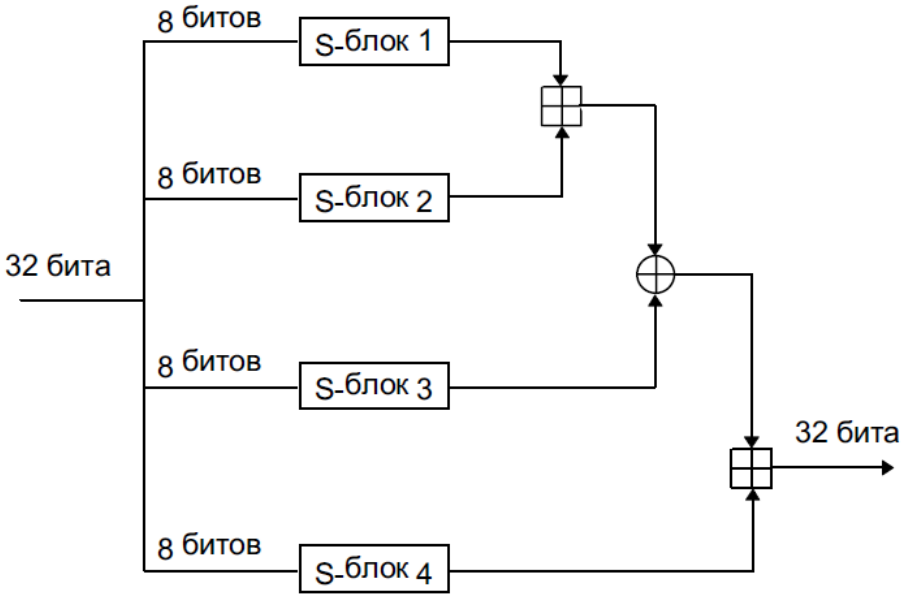


Рис. 3 Функция F

Процесс расширения ключа

1. Выбирается «искреннее число» (или иначе — «Nothing up my sleeve number»). Это такое число, которое изначально не содержит каких-либо повторяющихся последовательностей и является известным. Делается это для того, чтобы показать, что константа разработчиками была выбрана без преследования каких-то «гнусных» целей, наприимер для создания лазейки в алгоритме (backdoor). В качестве такого искреннего числа в Blowfish обычно используется число PI.

2. Значением мантиссы числа PI заполняется матрица раундовых ключей и матрицы подстановки

3. Значение каждого раундового ключа Pn (P1, P2 …) складывается по модулю 2 (XOR) с соответствующим элементами исходного ключа K. Например, выполняется XOR раундового ключа P1 с первыми 32 битами исходного ключа K, P2 со вторыми 32 битами

исходного ключа K и так далее. Если исходный ключ K короче длины всех раундовых ключей (576 бит), то он конкатенируется сам с собой: KK, KKK и так далее.

4. Далее необходимо зашифровать (вычислить новые значения) элементов матрицы раундовых ключей и матрицы подстановки. Для этого используется сеть Фейстеля для Blowfish.

   4.1. Используя текущие раундовые ключи P1—P18 и матрицы подстановок S1—S4, шифруем 64-битную последовательность нуля: 0x00000000 0x00000000, а результат записываем в P1 и P2.

   4.2. P1 и P2 шифруются изменёнными значениями раундовых ключей и матриц подстановки, результат записывается соответственно в P3 и P4.

   4.3. Шифрование продолжается до изменения всех раундовых ключей P1—P18 и элементов матриц подстановок S1—S4.

Шифрование/дешифрование

Процесс шифрации исходных данных строится по аналогии с процессом шифрации на этапе расширения ключа.

Дешифрование выполняется точно так же, как и шифрование, но P1- P18 используются в обратном порядке.

Режим OFB

Режим обратной связи вывода превращает блочный шифр в синхронный шифр потока: он генерирует ключевые блоки, которые являются результатом сложения с блоками открытого текста, чтобы получить зашифрованный текст. Так же, как с другими шифрами потока, зеркальное отражение в зашифрованном тексте производит зеркально отражённый бит в открытом тексте в том же самом местоположении. Это свойство позволяет многим кодам с исправлением ошибок функционировать как обычно, даже когда исправление ошибок применено перед кодированием.

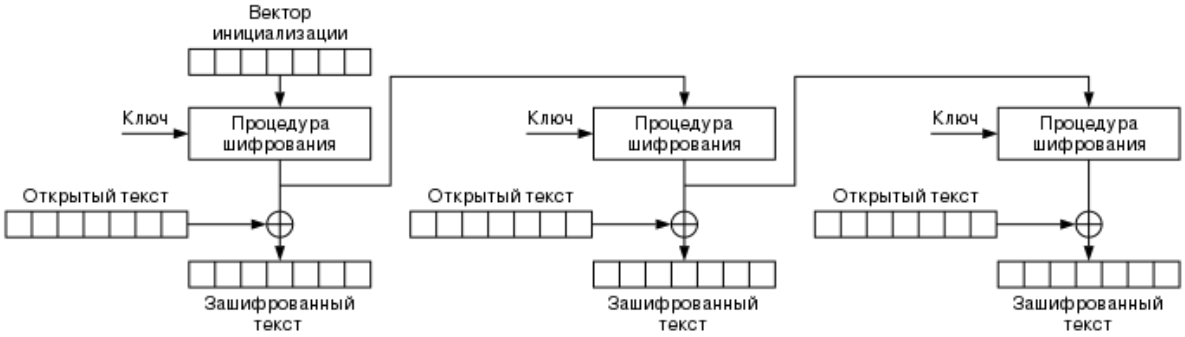


Рис. 4 Шифрование в режиме OFB

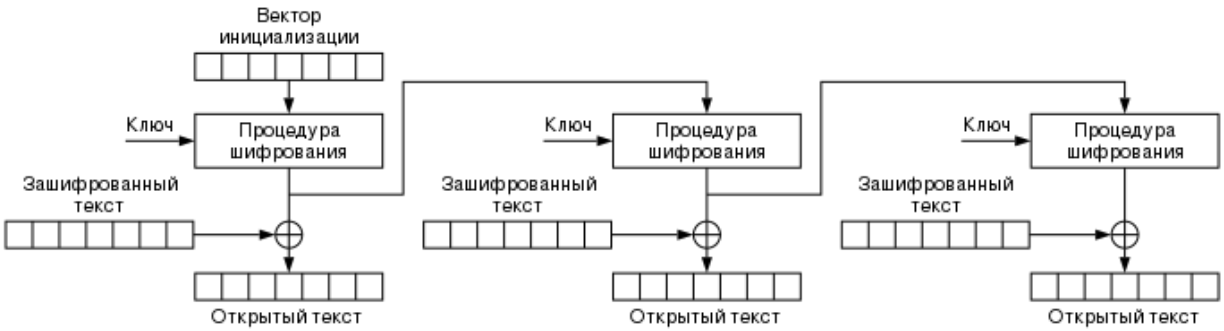
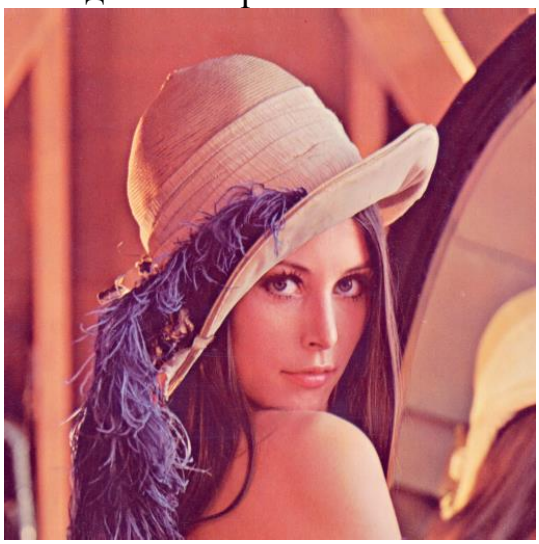


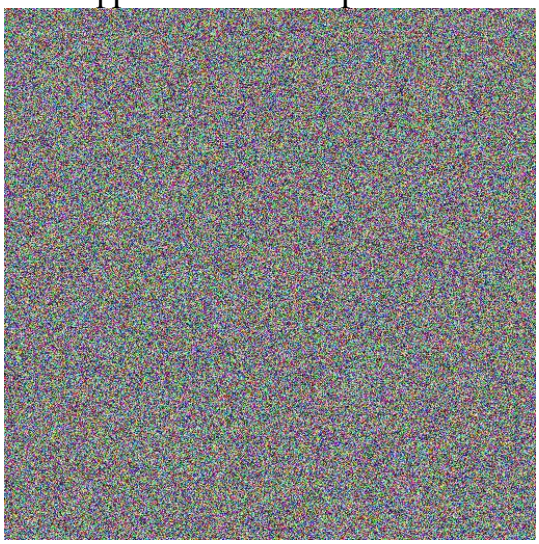
Рис. 5 Расшифрование в режиме OFB

3. Пример работы программы:

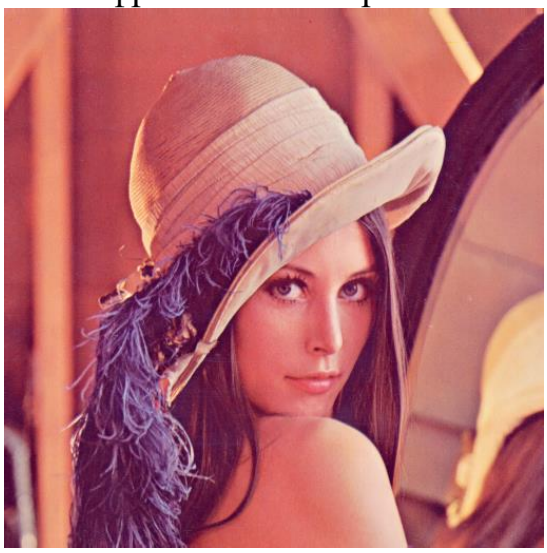
Исходное изображение:



Зашифрованное изображение:

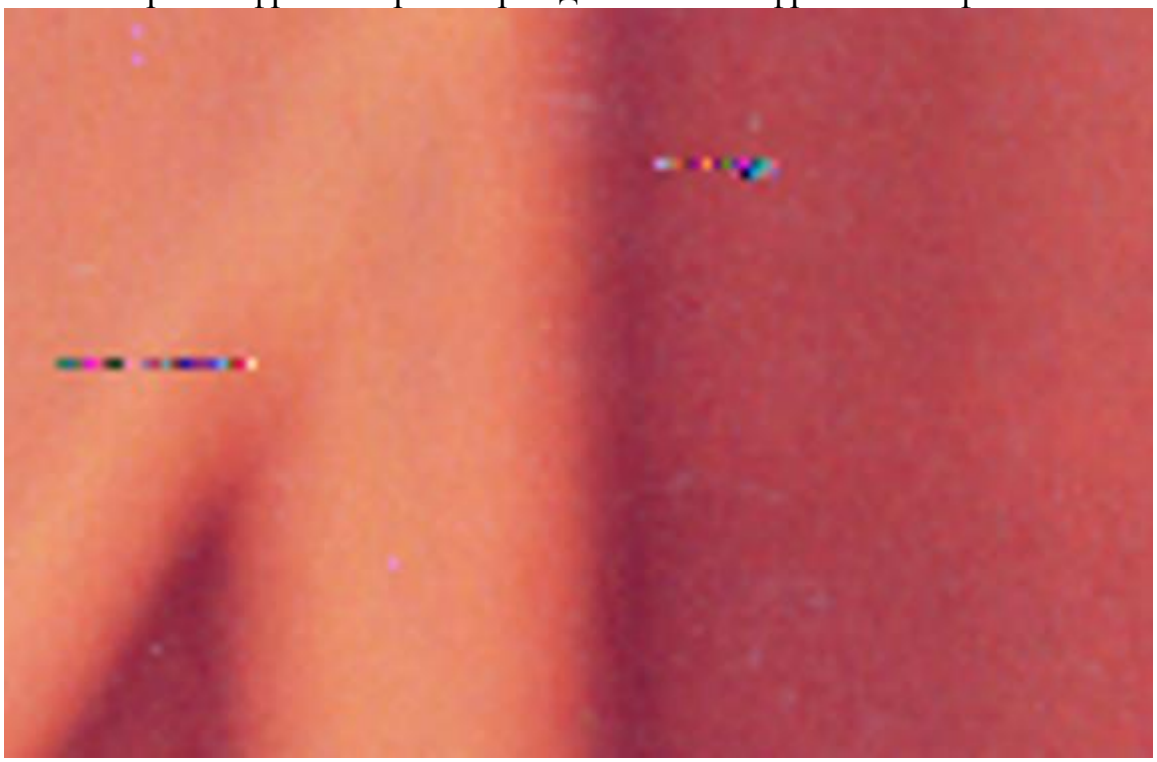


Расшифрованное изображение:



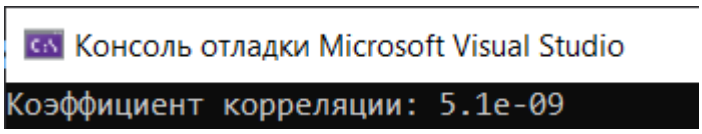
4. **Процесс распространения ошибок и коэффициент корреляции:**

Ошибки расшифровки при поврежденном зашифрованном файле:



Коэффициент корреляции:

$$\hat{r}_{A,B} = \frac{\hat{M}[(A - \hat{M}[A])(B - \hat{M}[B])]}{\hat{\sigma}_A \hat{\sigma}_B}$$



Коэффициент крайне мал, это говорит о том, что оригинальная и зашифрованная картинки очень сильно отличаются. Это нам и нужно, поскольку в таком случае дешифрация без знания ключа невозможна.

5. **Оценка размера пространства ключей:**

Длина ключа должна быть достаточно большой, чтобы предотвратить возможность перебора ключа.

В алгоритме Blowfish размер ключа варьируется от 64 бит до 448, следовательно понадобится перебрать от 2^{64} до 2^{448} вариантов, чтобы вскрыть данный шифр. Таким образом, Blowfish является надёжным шифром.

6. **Существующие атаки на алгоритм Blowfish:**

На данный момент известны две атаки на алгоритм.

Сдвиговая атака – не представляет опасности для Blowfish, так как была успешно проведена только на упрощенную версию алгоритма(было уменьшено количество раундов и убрана функция F). Позволяет вычислить использовались ли при шифровании слабые ключи или нет(но не позволяет узнать эти слабые ключи).

SWEET32 – атака позволяет взламывать 3DES и Blowfish, но срабатывает только при больших объёмах данных, измеряемых в гигабайтах(например 3DES удалось взломать, проанализировав 785 гб трафика).

7. **Вывод:**

В этой работе был реализован блочный алгоритм шифрования Blowfish с дополнительным режимом шифрования OFB с помощью языка C++.

Несмотря на обнаруженную атаку SWEET32, алгоритм все ещё достаточно хорош для работы с не слишком «тяжелыми» данными.

Огромный плюс – это его скорость(если не генерировать ключи каждый раз).

8. **Список литературы:**

1. Черчхаус. Коды и шифры

2. С.В. Беззатеев, Е.А. Крук, А.А. Овчинников “Блоковые Шифры. Учебное пособие”

3. М.Р. Гильмутдинов, А.М. Тюрликов, Е.М. Линский “Цифровая обработка изображений”