Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
«Сыктывкарский государственный университет имени Питирима Сорокина»
Институт точных наук и информационных технологий
КАФЕДРА ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Допустить к защите
Зав. кафедрой информационной безопасности,
к. ф. -м. н., доцент
_____ Л. С. Носов
«____»_____ 2016 года

Курсовая работа
по дисциплине «Информационные технологии»
Стандарты РФ на блочное шифрование.
Программная реализация алгоритма «Кузнечик».

Научный руководитель,
к. ф. -м. н., доцент _____ Ю. В. Гольчевский

Исполнитель, студент 133 группы _____ Е. Ю. Пипуныров

Сыктывкар, 2016 г.

Содержание

Список использованных определений

В настоящей работе применяются следующие термины с соответствующими определениями:

Алгоритм зашифрования (encryption algorithm) — алгоритм, реализующий зашифрование, т.е. преобразующий открытый текст в шифртекст.

Алгоритм расшифрования (decryption algorithm) — алгоритм, реализующий расшифрование, т.е. преобразующий шифртекст в открытый текст.

Базовый блочный шифр (basic block cipher) — блочный шифр, реализующий при каждом фиксированном значении ключа одно обратимое отображение множества блоков открытого текста фиксированной длины в блоки шифртекста такой же длины.

Блочный шифр (block cipher) — шифр из класса симметричных криптографических методов, в котором алгоритм зашифрования применяется к блокам открытого текста для получения блоков шифртекста.

Зашифрование (encryption) — обратимое преобразование данных с помощью шифра, которое формирует шифртекст из открытого текста.

Итерационный ключ (round key) — последовательность символов, вычисляемая в процессе развертывания ключа шифра, и определяющая преобразование на одной итерации блочного шифра.

Ключ (key) — изменяемый параметр в виде последовательности символов, определяющий криптографическое преобразование.

Развертывание ключа (key schedule) — вычисление итерационных ключей из ключа шифра.

Симметричный криптографический метод (symmetric cryptographic technique) — Криптографический метод, использующий один и тот же ключ для преобразования, осуществляемого отправителем, и преобразования, осуществляемого получателем.

Шифр (cipher) — криптографический метод, используемый для обеспечения конфиденциальности данных, включающий алгоритм зашифрования и алгоритм расшифрования.

Шифртекст (ciphertext) — данные, полученные в результате зашифрования открытого текста с целью скрытия его содержания.

S-блок (substitution box or S-box) — замещает маленький блок входных бит на другой блок выходных бит. Эта замена должна быть взаимно однозначной, чтобы гарантировать обратимость. Назначение S-блока заключается в нелинейном преобразовании, что препятствует проведению линейного криптоанализа.

P-блок (permutation box or P-box) — блок меняет местами все биты входных данных. Важным качеством P-блока является возможность распределить данные между входами как можно больших последующих блоков.

Список использованных обозначений

В настоящей работе применяются следующие обозначения:

Введение

В 2015 году в России были приняты новые стандарты на блочное шифрование в РФ - ГОСТ Р 34.12-2015 «Блочныйе шифры» и ГОСТ Р 34.13-2015 «Режимы работы блочных шифров». ГОСТ Р 34.12-2015 был принят взамен старого доброго ГОСТ 28147-89. В данном ГОСТ-е помимо старого алгоритма с длиной блока 64 бита, который назвали «Магма», был введён новый шифр с длиной блока уже 128 бит и длиной ключа 256 бит, который получил название «Кузнечик».

ГОСТ Р 34.13-2015 был принят взамен старого ГОСТ ИСО/МЭК 10116-93. Вотличие от старого, новый ГОСТ содержит дополнительные режимы шифровани, отсутсвовавшие в старом ГОСТ-е. Так же, оба ГОСТ-а имеют намного более приемлимое качество, более полную информацию об испонении и понятные примеры тестовых векторов.

В условиях новизны стандартов, было решено, что подробное изучение и программная реализация этих алгоритмов и режимов их работы станут актуальной задачей на момент выполнения работы.

Целью данной работы является программная реализация алгоритма блочного шифрования с длиной блока 128 бит «Кузнечик» в режимах работы «простой замены» и «обратной связи по шифртексту».

Постановка задачи

Для достидения поставленной цели необходимо решить следующие задачи:

- Изучение теоретической базы.
- Подробное изучение ГОСТ 34.12-2015 и ГОСТ 34.13-2015.
- Программная реализация выбранного алгоритма и режимов работы.

Выбор метода реализации задач:

Основной задачей данной работы является программная реализация выбранного алгоритма. Инструментом для решения основной задачи был выбран язык программирования СИ. Основной платформой для программной реализации была выбрана ОС Linux. Интерфейс взаимодействия с пользователем был выполнен в виде аргументов командной строки.

1. Теоретические основы

1.1. Методы построения блочных шифров

Воснове алгоритма «Кузнечик» лежат два основных метода посторенияблочных шифров: метод «сетей Фейстеля» и метод «SP-сетей», предложенные в 1971 году Хорстом Фейстелем. Рассмотри подробнее эти методы.

1.1.1. SP-сеть

SP-сеть (Substitution-Permutation network, подстановочно-перестановочная сеть) — разновидность блочного шифра, предложенная в 1971 году Хорстом Фейстелем. В простейшем варианте представляет собой «сэндвич» из слоёв двух типов, используемых многократно по очереди. Первый тип слоя — P-слой, состоящий из P-блока большой разрядности, за ним идёт второй тип слоя — S-слой, представляющий собой большое количество S-блоков малой разрядности, потом опять P-слой и т. д. Первым криптографическим алгоритмом на основе SP-сети был «Люцифер» (1971). В настоящее время из алгоритмов на основе SP-сетей широко используется AES (Rijndael).

Рассмотрим работу сети на простом примере, содержащем всего 3 раунда(Рис.1) Шифр на основе SP-сети получает на вход блок и ключ и совершает несколько чередующихся раундов, состоящих из чередующихся стадий подстановки (S-блок) и стадий перестановки (P-блок). Для достижения безопасности достаточно одного S-блока, но такой блок будет требовать большого объёма памяти. Поэтому используются маленькие S-блоки, смешанные с P-блоками. Нелинейная стадия подстановки перемешивает биты ключа с битами открытого текста, создавая конфузию Шеннона. Линейная стадия перестановки распределяет избыточность по всей структуре данных, порождая диффузию[1].

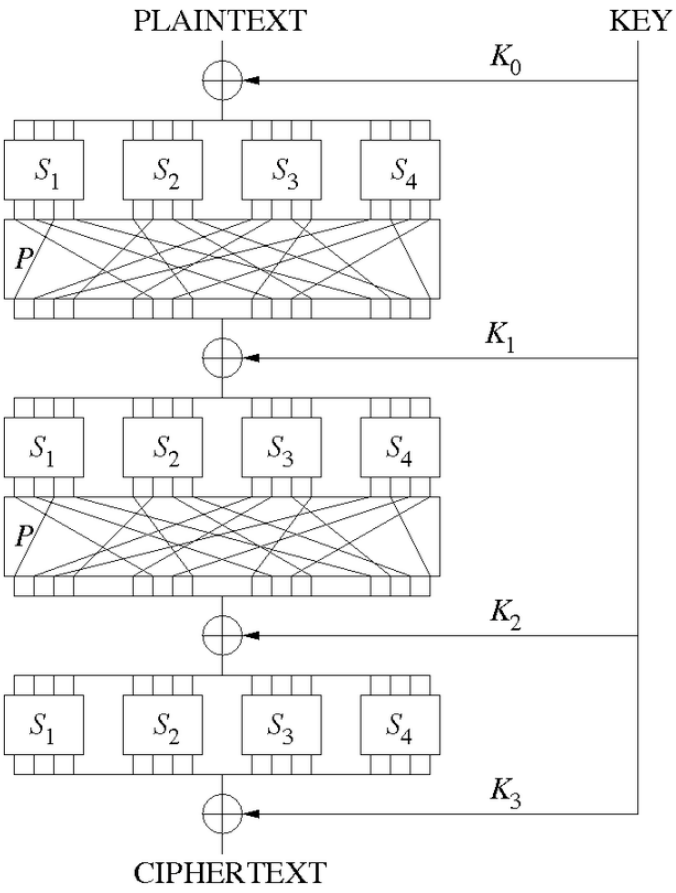

Рис. 1. Пример SP-сети с 3 раундами

1.1.2. Сеть Фейстеля

Сеть Фейстеля(Feistel network) — сеть, состоящая из ячеек, называемых ячейками Фейстеля. На вход каждой ячейки поступают данные и ключ. На выходе каждой ячейки получают изменённые данные и изменённый ключ. Все ячейки однотипны, и говорят, что сеть представляет собой определённую многократно повторяющуюся (итерированную) структуру. Ключ выбирается в зависимости от алгоритма шифрования/расшифрования и меняется при переходе от одной ячейки к другой. При шифровании и расшифровании выполняются одни и те же операции, отличаясь только порядком ключей. Ввиду простоты операций сеть Фейстеля легко реализовать как программно, так и аппаратно. Большинство современных блочных шифров (DES, RC2, RC5, RC6, Blowfish, FEAL, CAST-128, TEA, XTEA, XXTEA и др.) используют сеть Фейстеля в качестве основы[1].

Расммотрим работу сети Фейстеля на простом примере зашифрования блока текста(Рис.2). Алгоритм состоит из повторяющихся раундов, в каждом из которых выполняются следующие действия:

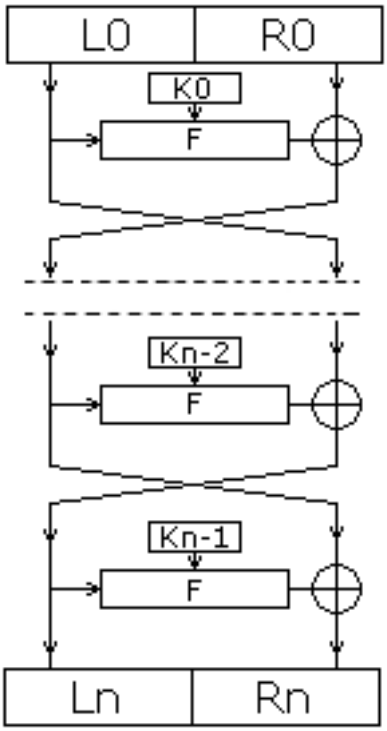

Рис. 2. Пример шифрования с использованием сети Фейстеля

1. Входной блок разбивается на левую (L) и правую (R) части.
2. Левая часть и раундовый ключ (K) поступают на вход функции шифрования(F).
3. Правая часть складывается по модулю 2 с результатом работы функции шифрования.
4. Происходит конкатенация обеих ветвей и циклический сдвиг вправо.

1.2. Конечные поля

Конечное поле, или поле Галуа — поле, состоящее из конечного числа элементов. Число элементов в поле называется его порядком. Конечное поле обычно обозначается GF(q) (сокращение от Galois field) и называется полем Галуа порядка q. С точностью до изоморфизма конечное поле полностью определяется его порядком, который всегда является степенью какого-нибудь простого числа, то есть $q = p^n$, где p — простое число, называемое так же основанием поля, а n — любое натуральное число. При этом p будет являться характеристикой этого поля. Понятие конечного поля используется, в теории чисел, теории групп, алгебраической геометрии, криптографии.

Элементы поля Галуа можно складывать и умножать и т.д., но эти операции отличаются от тех, которые используются для чисел. В данном параграфе будут описаны основные

математические операции в полях Галуа.

1.2.1. Сложение и вычитание

Сложение и вычитание 2 элементов в поле Галуа достигается за счет сложения и вычитания коэффициентов полиномов по модулю его основания соответственно. В дальнейшем все операции будут выполняться в поле GF(2^8). Таким образом, в рамках данной работы, они эквивалентны операции XOR: $1 \oplus 1 = 0, 1 \oplus 0 = 1, 0 \oplus 0 = 0$.

Рассмотри для примера операцию сложения. Добавление конечных элементов поля Галуа можно охарактеризовать как сложение по модулю 2 соответствующих битов в байте. Для 2 байтов $[a_7, a_{6,} a_5, a_4, a_3, a_2, a_1, a_0]$, и $[b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0]$ суммой будет являться байт $[c_7, c_{6,} c_5, c_4, c_3, c_2, c_1, c_0]$, где $c_i = a_i \oplus b_i$.

Следующие выражения эквивалентны друг другу:

$(x_6 + x_4 + x_2 + x + 1) + (x_7 + x + 1) = x_7 + x_6 + x_4 + x_2$ *(Полиноминальная запись)*

$[01010111] \oplus [10000011] = [11010100]$ *(Бинарная запись)*

$[57] \oplus [83] = [d4]$ *(Шестнадцатиричная запись)*

1.2.2. Умножение

В полиномиальном представлении, умножение в поле Галуа GF(2^8) (обозначается $\times$) соответствует умножению полиномов по модулю неприводимого полинома степени 8(обозначим $m(x)$). Полином называется неприводимым, если его нельзя разложить на нетривиальные (неконстантные) многочлены. Например, $[57] \times [83] = [c1]$, потому что:

$$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) =$$
$$= x^{13} + x^{11} + x^9 + x^8 + x^7 + x^5 + x^3 + x^2 + x + x^6 + x^4 + x^2 + x + 1 =$$
$$= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$$

и $(x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1) \mod (x^8 + x^4 + x^3 + x + 1) = x^7 + x^6 + 1$

Сокращение на $m(x)$ гарантирует, что результатом будет полином степени меньше 8, и он может быть представлен байтом.

Операция деления заключается в следующем:

Для любого ненулевого двоичного полинома b(x) степени меньше 8, можно найти обратный многочлен b^{-1}(x), с помощью расширеного алгоритма Евклида, используя полиномы $a(x)$ и $c(x)$:

$$b(x)a(x) + m(x)c(x) = 1 \qquad (1.1)$$

Но $a(x) \times b(x) \bmod m(x) = 1$ значит:

$$b^{-1}(x) = a(x) \bmod m(x).$$

Для упрощения и ускорения операций умножения и деления, составляют специальные логарифмические таблицы, исходя из следующего равенства:

$$ab = g^{\log_g(ab)} = g^{\log_g a + \log_g b}$$

Где g выбирается из т.н.«генерирующих» полиномов поля. Подобноые полиномы являются частями всех конечных полей. Их главное свойство — неприводимость[2].

2. ГОСТ-Р 34.12-2015. Алгоритм «Кузнечик»

Для решения поставленной задачи, в данной работе будет рассмотрен лишь алгоритм с длиной блока 128 бит, получивший название «Кузнечик»". В отличие от ГОСТ 28147-89 новый шифр представляет собой не сеть Фейстеля, а SP-сеть. Шифрование основано на последовательном применении нескольких однотипных раундов, каждый из которых содержит три преобразования: сложение с раундовым ключом, преобразование блоком подстановок и линейное преобразование. Для того, чтобы понять работу алгоритма, разберём все преобразования, применяемые к блоку входного текста на каждом раунде на конкретном примере.

2.1. Преобразования

Все преобразования будут сопровождаться примерами со следующими параметрами:

a = 1122334455667700ffeeddccbbaa9988 — входной блок текста,

k = 99BB99FF99BB99FFFFFFFFFFFFFFFFFF — раундовый ключ.

Вначале 128-битный входной вектор очередного раунда складывается побитно с раундовым ключом(Рис.3):

$$X[k](a) = k \oplus a, \text{ где}$$

k — раундовый ключ,

a — входной блок текста.

Затем результат подвергается нелинейному биективному преобразованию, таблица для которого определена в ГОСТ-е. Работает оно следующим образом:

128-битный вектор после сложения по модулю два побайтно преобразовывается в десятичный вид, тем самым определяется позиция байта в таблице подстановок (S-блок), затем с этой позиции считывается число в десятичном виде и преобразуется назад в шестнадцатеричный вид. Например, шестнадцатеричному числу 99 соответствует десятичное 153. Элемент с номером 153 в таблице подстановок имеет значение 232, что соответствует шестнадцатеричному E8(Рис.4). Данное преобразование обозначено в ГОСТ-е как S.

$$S(a) = S(a_{15}\|\cdots\|a_0) = \pi(a_{15})\|\cdots\|\pi(a_0),$$

где $a = a_{15}\|\cdots\|a_0$, π — операция подстановки.

Следующим этапом зашифрования является линейное преобразование, выполняемое с использованием линейного регистра сдвига с обратной связью. Работает оно следующим образом: очередной байт входного юлока считывается, затем складывается по модулю два с результатами умножаения других байтов входного блока в поле на коэффициенты 148, 32, 133, 16, 194, 192, 1, 251, 1, 192, 194, 16, 133, 32, 148, 1 в зависимости от номера байта. Регистр сдвигается в сторону младшего байта на один байт. Полученное число в шестнадцатеричном

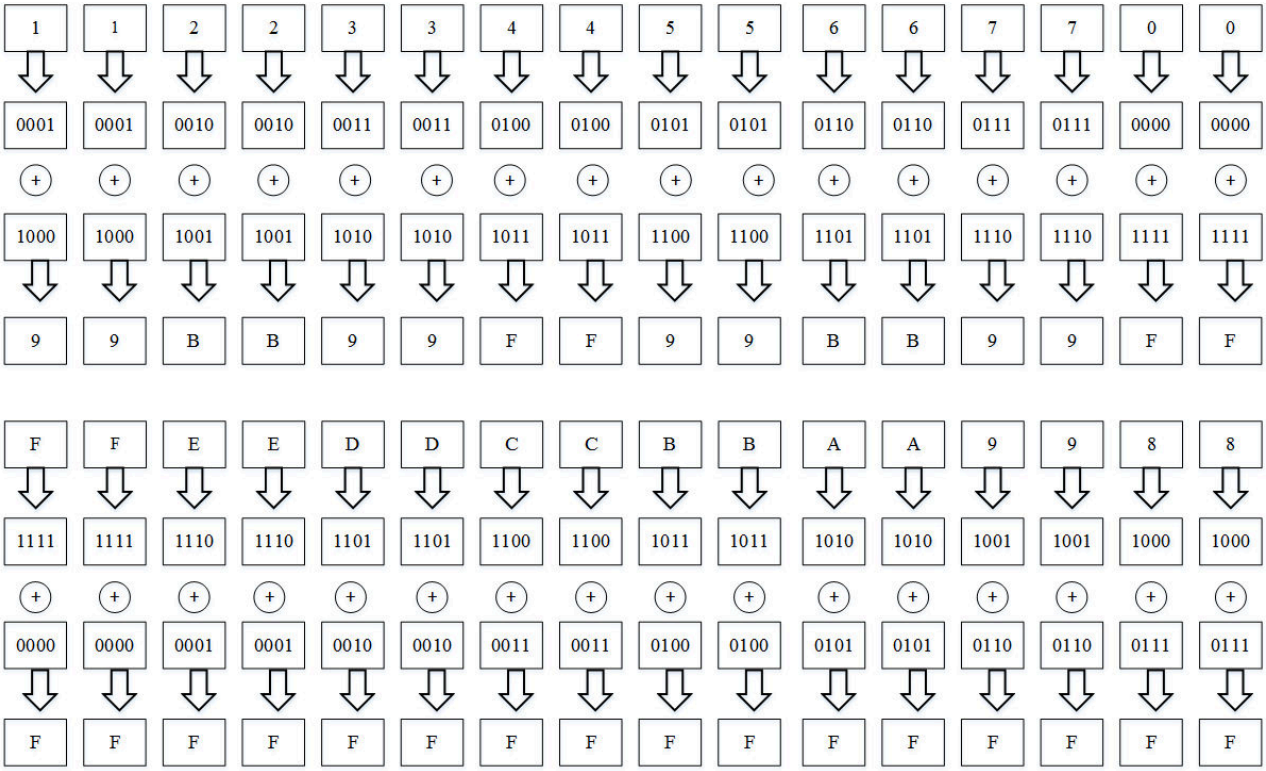

Рис. 3. Пример X-преобразования

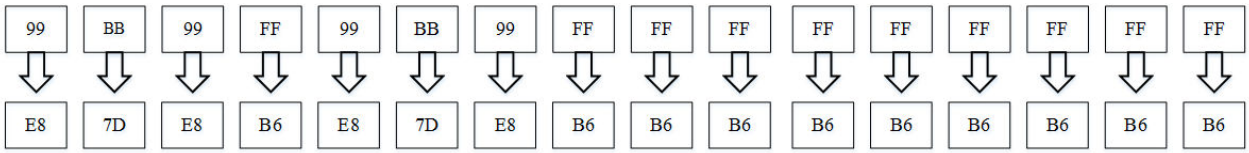

Рис. 4. Пример S-преобразования

виде записывается на место старшего байта. Регистр сдвигается и перезаписывается 16 раз. Данное преобразование обозначено в ГОСТ-е как S. Схематическое изображение работы L-преобразования показано на рисунке 5, а пример его работы на рисунке 6.

$$R(a) = R(a_{15} \| \cdots \| a_0) = l(a_{15} \cdots a_0) a_{15} \| \cdots \| a_1,$$

$$L(a) = R^{16}(a)$$

где $a = a_{15} \| \cdots \| a_0$ — блок входного тескта

Результатом L-преобразования будет следующий 128-битный вектор $a = E297B686E355B0A1CF4A2F9249140830$. Это результат работы первого раунда алгоритма, таким же образом будут проходить последующие 8 раундов, результаты их работы

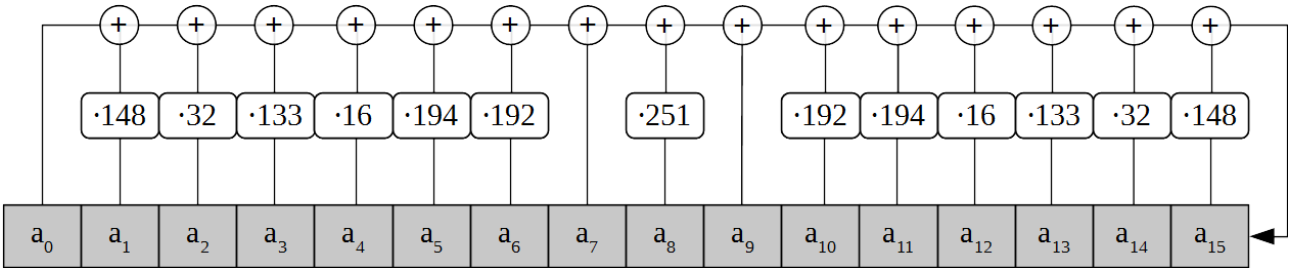

Рис. 5. Схема L-преобразования

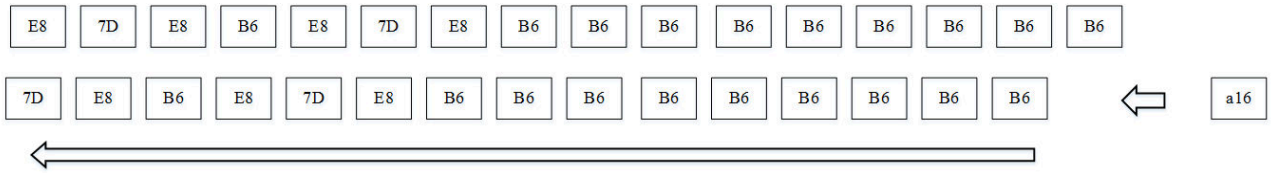

Рис. 6. Пример L-преобразования

можно посмотреть в описании стандарта. Схематическое изображение одного раунда можно увидеть на рисунке 7. Заключительный 10 раунд включает в себя только X-преобразование

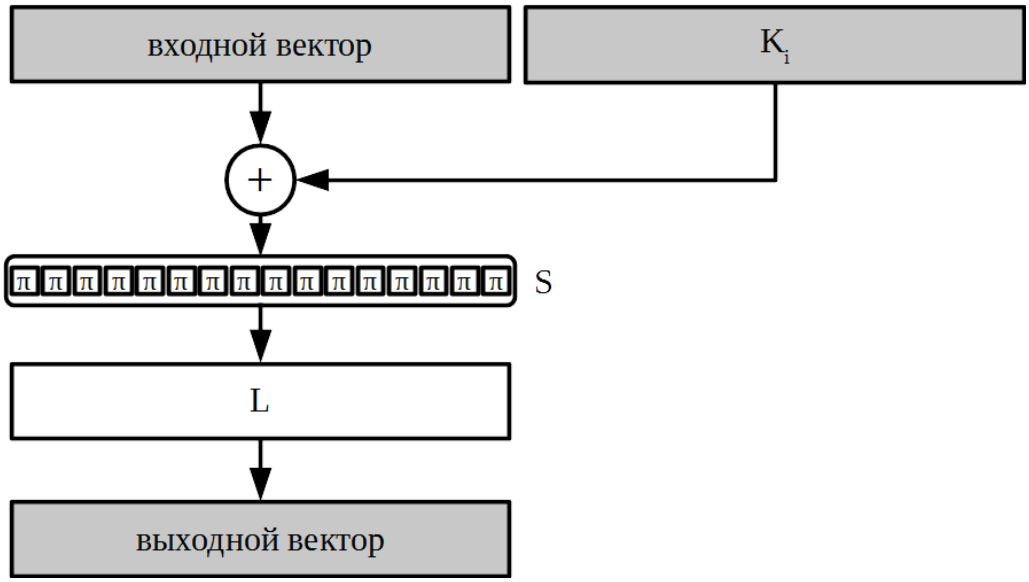

Рис. 7. Схема одного раунда преобразования

результатов работы 9 раундов и ключа 10 раунда.

2.2. Выработка раундовых ключей

Рассмотрим теперь процедуру генерации раундовых ключей из мастер-ключа. Первые два получаются разбиением мастер-ключа пополам. Далее для выработки очередной пары раундовых ключей используется 8 итераций сети Фейстеля, где, в свою очередь, в качестве раундовых ключей используется счетчиковая последовательность, прошедшая через линейное (L) преобразование алгоритма:

$$F[k](a_1, a_0) = (LSX[k](a_1) \oplus a_0, a_1),$$

где k — раундовый ключ алгоритма развёртки ,

a_0, a_1 — входныеблокитекстадляочередногораундаалгоритмаразвёртки

$$C_i = L(i),\ i = 1, 2, \ldots, 32.$$

$$(K_{2i+1},\ K_{2i+2}) = F[C_{8(i-1)+8}] \cdots F[C_{8(i-1)+1}](K_{2i-1},\ K_{2i}),\ i = 1, 2, 3, 4.$$

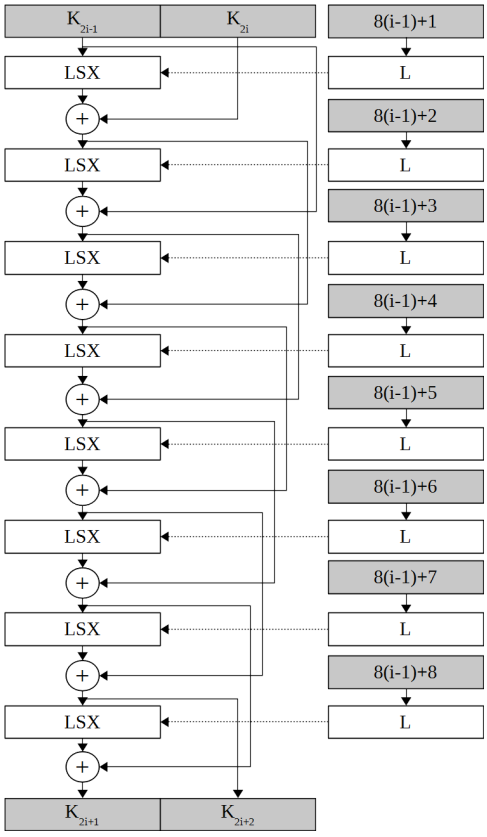

Рис. 8. Схема одного раунда алгоритма развёртки ключей

Раунд ключевой развертки можно изображён на рисунке 8:

А вся процедура выработки пары раундовых ключей изображена на рисунке 9.

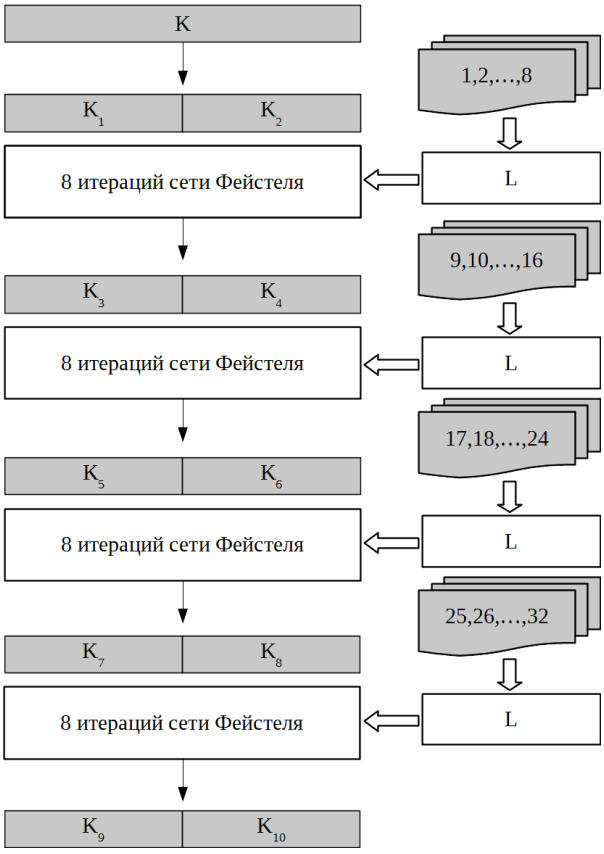

Рис. 9. Схема выработки очередной пары раундовых ключей

2.3. Шифрование и расшифрование

В результате, шифрование одного 128-битного входного блока описывается следующим уравнением:

$$E_{K_1 \cdots K_{10}}(a) = X[K_{10}]LSX[K_9]\cdots LSX[K_1](a)$$

В виде блок-схемы шифрование представлено на рисунке 10.

Расшифрование реализуется обращением базовых преобразований и применением их в обратном порядке[3]:

$$D_{K_1 \cdots K_{10}}(a) = X[K_1]S^{-1}L^{-1}X[K_2]\cdots S^{-1}L^{-1}X[K_{10}](a)$$

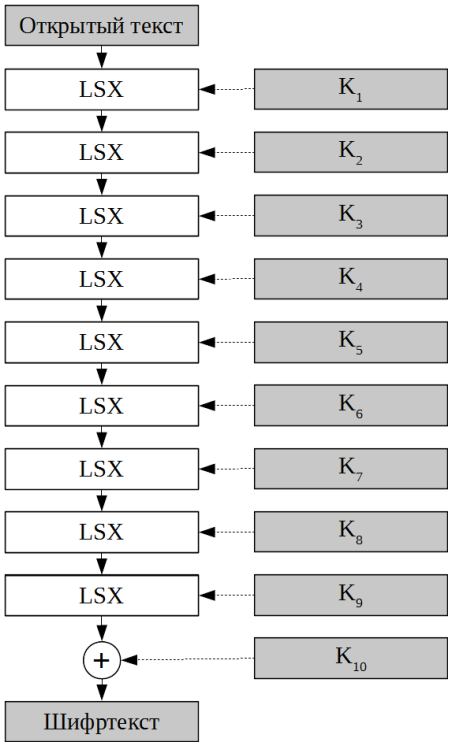

Рис. 10. Схема шифрования одного блока

3. ГОСТ 34.13-15 «Режимы работы блочных шифров»

Данный стандарт содержит в себе описание основных режимов работы современных блочных шифров, а так же используемые в реализации дополнительные операции.

В стандарте используются следующие термины с соответствующими определениями:

базовый блочный шифр (basic block cipher) — блочный шифр, реализующий при каждом фиксированном значении ключа одно обратимое отображение множества блоков открытого текста фиксированной длины в блоки шифртекста такой же длины.

дополнение (padding) — приписывание дополнительных бит к строке бит.

зацепление блоков (block chaining) — шифрование информации таким образом, что каждый блок шифртекста криптографически зависит от предыдущего блока шифртекста.

зимитовставка (message authentication code) — строка бит фиксированной длины, полученная применением симметричного криптографического метода к сообщению, добавляемая к сообщению для обеспечения его целостности и аутентификации источника данных.

синхропосылка (initializing value) — комбинация знаков, передаваемая по каналу связи и предназначенная для инициализации алгоритма шифрования.

счётчик (counter) — Строка бит длины, равной длине блока блочного шифра, используемая при шифровании в режиме гаммирования.

3.1. Вспомогательные операции

3.1.1. Дополнение сообщения

Некоторые режимы работы блочных шифров требуют,чтобы длина сообщения была кратна некоторой величине l(длина входного блока). В последнем случае при работе с сообщениями произвольной длины необходимо применение процедуры дополнения сообщения до требуемой длины. Ниже приведены три процедуры дополнения[4]. P в дальнейшем обозначает исходное собщение, P^*.

Процедура 1

Пусть $r = |P| \bmod l$, тогда:

$$P^* = \begin{cases} P, \text{ если } r = 0 \\ P \| 0^{l-r}, \text{ иначе} \end{cases}$$

Описанная процедура в некоторых случаях не обеспечивает однозначного восстановления исходного сообщения, а потому, для однозначного восстановления необходимо дополнительно знать длину исходного сообщения[4].

Процедура 2

Пусть $r = |P| \bmod l$, тогда:

$$P^* = P \| 1 \| 0^{l-r-1}.$$

Данная процедура обеспечивает однозначное восстановление исходного сообщения. При этом если длина исходного сообщения кратна l, то длина дополненного сообщения будет увеличена на длину блока шифра[4].

Процедура 3

Пусть $r = |P| \bmod l$, тогда: В зависимости от значения r возможны случаи:

- если $r = n$, то последний блок не изменяется $P^* = P$
- если r < n, то применяется процедура 2.

Данная процедура обязательна для режима выработки имитовставки и не рекомендуется для использования в других режимах[4].

3.1.2. Синхропосылка

В некоторых режимах работы используются величины, начальное значение которых вычисляется на основании синхропосылки IV; Во всех описываемых в настоящем стандарте режимах работы не требуется обеспечение конфиденциальности синхропосылки. Вместе с тем процедура выработки синхропосылки должна удовлетворять одному из следующих требований.

- Значения синхропосылки для режимов простой замены с зацеплением и гаммирования с обратной связью по шифртексту необходимо выбирать случайно, равновероятно и независимо друг от друга из множества всех допустимых значений. В этом случае значение каждой используемой синхропосылки IV должно быть непредсказуемым (случайным или псевдослучайным): зная значения всхе других используемых синхропосылок, значение IV нельзя определить с вероятностью большей, чем $2^{-|IV|}$.
- Все значения синхропосылок, выработанных для зашифрования на одном и том же ключе в режиме гаммирования, должны быть уникальными, т.е. попарно различными. Для выработки значений синхропосылок может быть использован детерминированный

счетчик.

- Значение синхропосылки для режима гаммирования с обратной связью по выходу должно быть либо непредсказуемым (случайным или псевдослучайным), либо уникальным[4].

3.1.3. Процедура усечения

В некоторых режимах используется усечение строк длины n до строк длины s. В таких случаях используется функция T_s, закюлающаяся во взятии бит с большими номерами[4].

3.2. Режимы работы алгоритмов блочного шифрования

В рамках данной работы будут рассмотрены только два режима работы шифров, реализованные программно:

- Режим простой замены
- Режим гаммирования с обратной связью по шифртексту

Описание остальных режимов работы Вы можете глянуть в приложенном ГОСТ 34.13-15.

Далее будут употребляться следующие обозначения:

e_k — базовый алгоритм зашифрования.

d_k — базовый алгоритм расшифрования.

3.2.1. Режим простой замены

Длина сообщений, зашифровываемых в режиме простой замены, должна быть кратна длине блока базового алгоритма блочного шифрования n, поэтому, при необходимости, к исходному сообщению должна быть предварительно применена процедура дополнения. Зашифрование (расшифрование) в режиме простой замены заключается в зашифровании (расшифровании) каждого блока текста с помощью базового алгоритма блочного шифрования[4].

Зашифрование

Открытый и, при необходимости, дополненный текст $|P| = n * q$ представляется в виде: $P = P_1 \| \cdots \| P_i \| \cdots \| P_q$, $i = 1, 2, \ldots, q$. Блоки шифртекста вычисляются по следующему правилу:

$$C_i = e_k(P_i),\ i = 1, 2, .., q.$$

Результирующий шифртекст имеет вид:

$$C = C_1 \| C_2 \| \ldots \| C_q.$$

Зашифрование в режиме простой замены проиллюстрировано на рисунке 11

Расшифрование

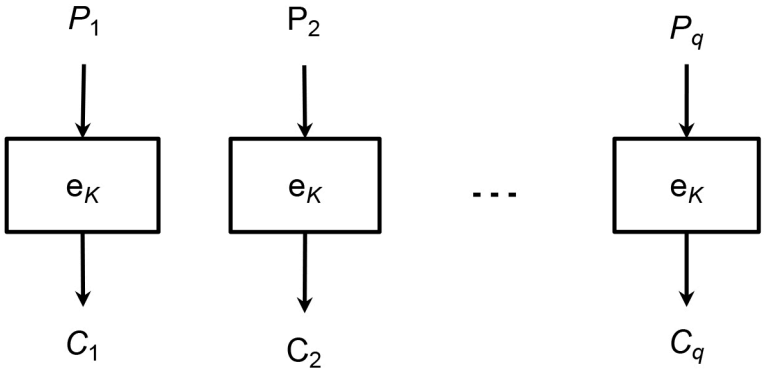

Рис. 11. Схема зашифрования в режиме простой замены

Шифртекст представляется в виде: $C = C_1 \| C_2 \| \ldots \| C_q$.Блоки открытого текста вычисляются по следующему правилу:

$$P_i = d_k(C_i),\ i = 1, 2, .., q.$$

Исходный (дополненный) открытый текст имеет вид:

$$P = P_1 \| P_2 \| \ldots \| P_{q.}$$

Если к исходному открытому тексту была применена процедурадополнения, то после расшифрования следует произвести обратную процедуру. Для однозначного восстановления сообщения может потребоваться знание длины исходного сообщения[4].

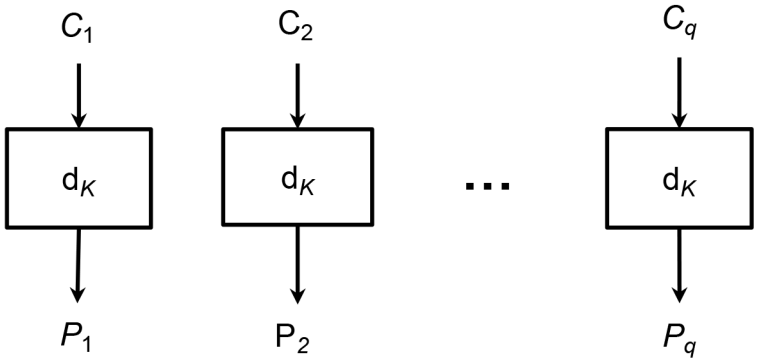

Рис. 12. Схема расшифрования в режиме простой замены

Расшифрование в режиме простой замены проиллюстрировано на рисунке 12.

3.2.2. Режим гаммирования с обратной связью по шифртексту

Параметрами режима гаммирования с обратной связью по шифртексту являются целочисленные величины s и m , $0 < s \le n$, $n \le m$.

В конкретной системе обработки информации на длину сообщения P может как накладываться ограничение $|P| = s * q$, так и не накладываться никаких ограничений. В случае если такое ограничение накладывается, к исходному сообщению, при необходимости, должна быть предварительно применена процедура дополнения[4].

При шифровании на одном ключе для каждого отдельного открытого текста используется значение непредсказуемой (случайной или псевдослучайной) синхропосылки IV.

При шифровании в режиме гаммирования с обратной связью по шифртексту используется двоичный регистр сдвига R длины m. Начальным заполнением регистра является значение синхропосылки IV.

Зашифрование в режиме гаммирования с обратной связью по шифртексту заключается в покомпонентном сложении открытого текста с гаммой шифра, которая вырабатывается блоками длины s. При вычислении очередного блока гаммы выполняется зашифрование n разрядов регистра сдвига с большими номерами базовым алгоритмом блочного шифрования с последующим усечением. Затем заполнение регистра сдвигается на s разрядов в сторону разрядов с большими номерами, при этом в разряды с меньшими номерами записывается полученный блок шифртекста, являющийся результатом покомпонентного сложения гаммы шифра и блока открытого текста[4].

Зашифрование

Открытый текст P представляется в виде $P = P_1 \| P_2 \| \ldots \| P_q$, Блоки шифртекста вычисляются по следующему правилу:

$$R_1 = IV,$$

$$\begin{cases} C_i = P_i \oplus T_s(e_k(MSB_n(R_i))), \\ R_{i+1} = LSB_{m-s}(R_i) \| C_i, \end{cases}$$

$$i = 1, 2, \ldots, q-1,$$

$$C_q = P_q \oplus T_r(e_k(MSB_n(R_q))),$$

Результирующий шифртекст имеет вид:

$$C = C_1 \| C_2 \| \ldots \| C_q.$$

Зашифрование в режиме гаммирования с обратной связью по шифртексту проиллюстрировано на рисунке 13.

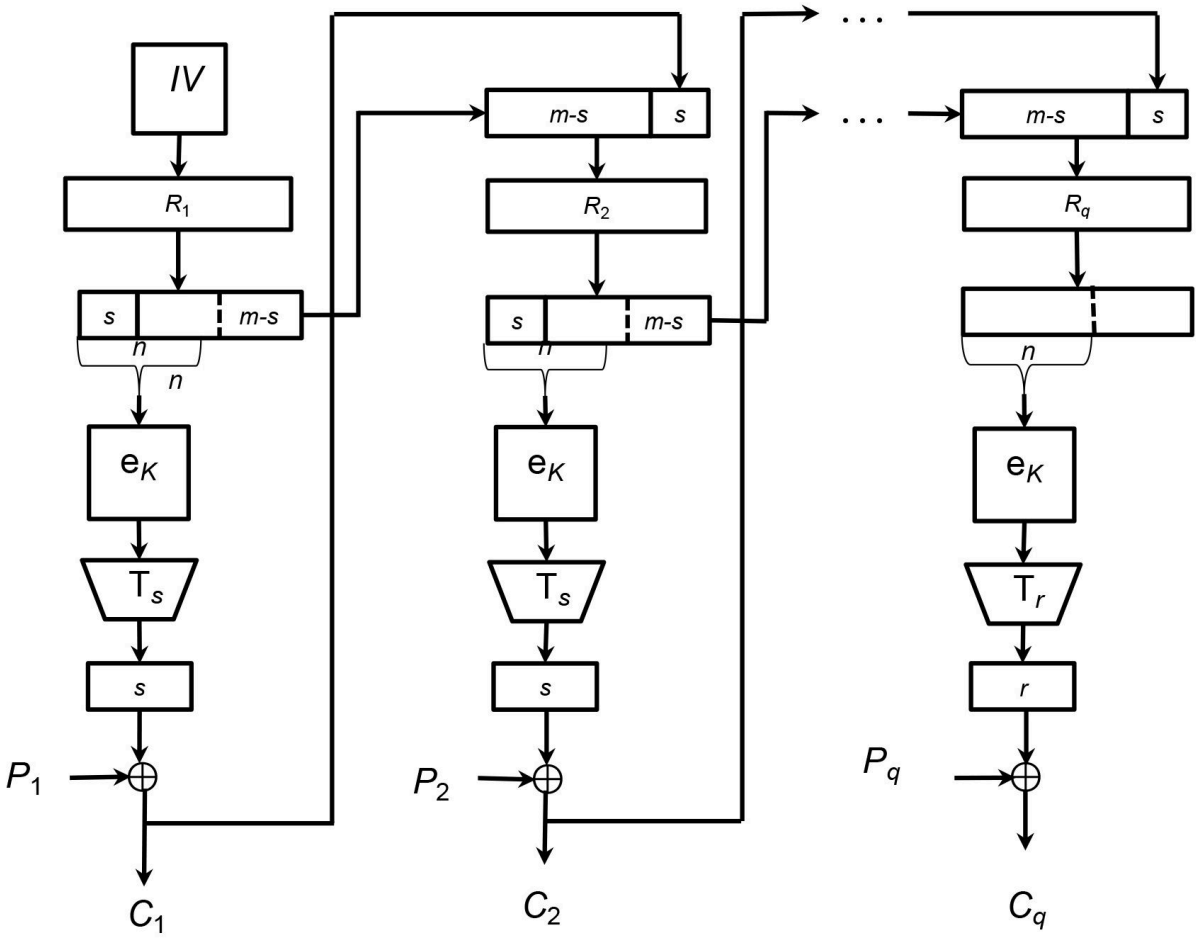

Рис. 13. Схема зашифрования в режиме гаммирования с обратной связью по шифртексту

Расшифрование

Шифртекст представляется в виде: $C = C_1 \| C_2 \| \ldots \| C_q$. Блоки открытого текста вычисляются по следующему правилу:

$$R_1 = IV,$$

$$\begin{cases} P_i = C_i \oplus T_s(e_k(MSB_n(R_i))), \\ R_{i+1} = LSB_{m-s}(R_i) \| C_{i,} \end{cases}$$

$$i = 1, 2, .., q-1,$$

$$P_q = C_q \oplus T_r(e_k(MSB_n(R_q))),$$

Исходный открытый текст имеет вид:

$$P = P_1 \| P_2 \| \ldots \| P_q.$$

Если к исходному открытому тексту была применена процедура дополнения, то после расшифрования следует произвести обратную процедуру. Для однозначного восстановления сообщения может потребоваться знание длины исходного сообщения[4].

Расшифрование в режиме гаммирования с обратной связью по шифртексту проиллюстри-

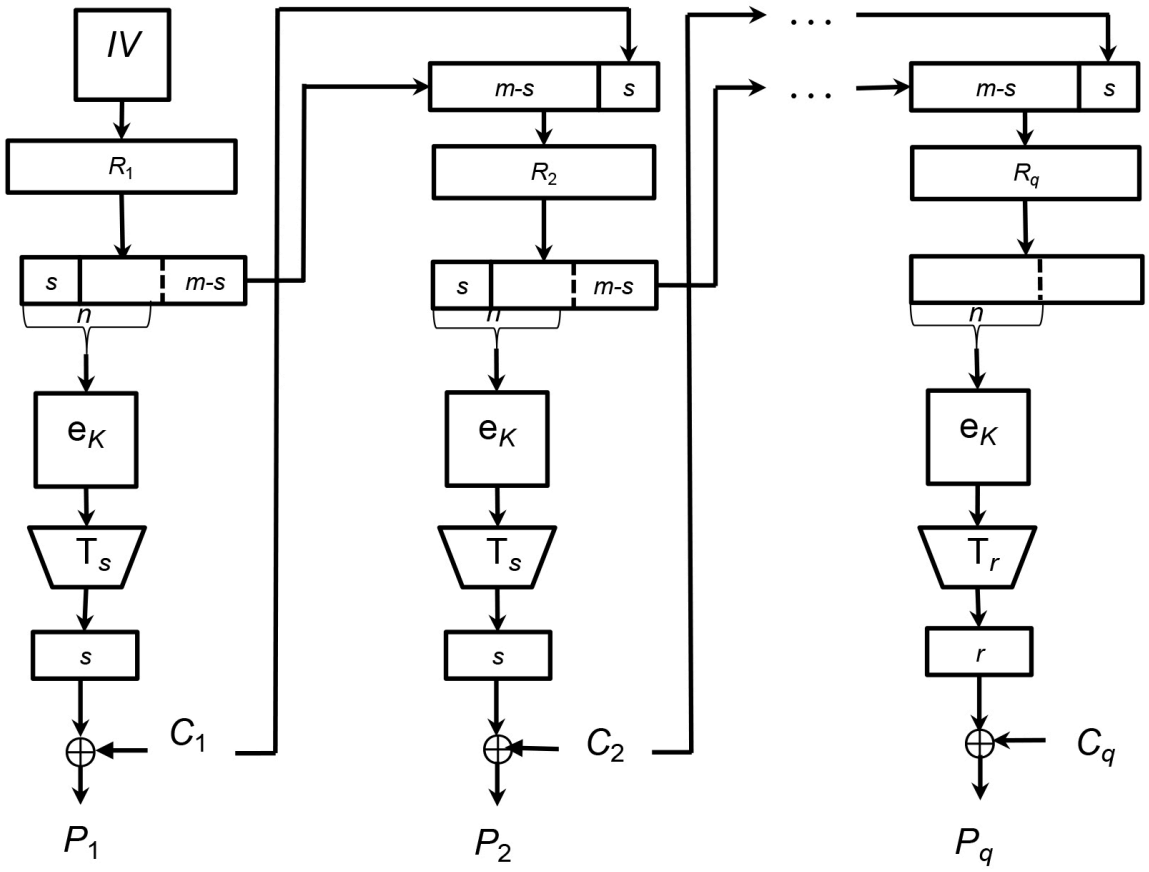

Рис. 14. Схема расшифрования в режиме гаммирования с обратной связью по шифртексту

ровано на рисунке 14.

4. Программная реализация

В конечном итоге была написана программа, код которой приведён в приложении к данной работе. Интерфейсом программы, как было отмечено выше, выбраны аргументы командной строки. Ниже приведён вывод вспомогательного сообщения при использовании недекларированного флага.

```
Usage: kuzn_chiper [OPTIONS] [FILE]
Transform standard input or predefined input file by GOST 34.12-15
to standard output or predefined output file.

Options:
  -e <encrypt> encrypt input
  -d <decrypt>          decrypt input
  -k <secret-key-file>  secret key file(hex)
  -s <simple mode>      work in simple mode
  -t <OFB mode>       work in Outtext FeedBack mode
  with inital gamma synchronisation file (hex)
  -s <sinple mode>
  -i <input-file>       input file
  -o <output-file>      output file

  If no FILE provided, then reads standard input.

Examples:
  kuzn_chiper -e -k key -t sync -i /etc/passwd > passwd.encr
```

Вывод вспомогательного сообщения

Из вспомогательного сообщения видно, что режим работы шифратора однозначно выставляется с помошью соответствующих аргументов. Единственным обязательным аргументом является указание файла с ключом. Остальные аргументы имеют стандартные значения. Стандартно шифратор работает в режиме простой замены, исходный текст читается со стандартного ввода, а шифртекст записывается в станартный вывод.

Структурно программа состоит из центрального файла `main.c`, в котором обрабатываются аргументы программы, файла `kuzn.c`, в котором реализованы все функции шифорвания текста и файла `gf.c`, в котором реализованы все испоьзующиеся функции работы с полем Галуа.

В заголовочном файле kuzn.h находятся объявления использующиихся функций из файла kuzn.c.

```
void Xfunc(uint8_t* block, uint8_t* raund_key);
void Cfunc(int i, uint8_t *C);
void Ffunc(uint8_t *C, int i, uint8_t[10][16]);
void efunc(uint8_t *block, uint8_t[10][16]);
void dfunc(uint8_t *block, uint8_t[10][16]);

int bl_gen_en(uint8_t *block, FILE *input_f);
int bl_gen_de(uint8_t *block, FILE *input_f);
int sync_gen(uint8_t *sync, FILE *sync_f);
int keys_gen(uint8_t[10][16], FILE *key_f);

int simple_mode_en(uint8_t [10][16], FILE *input_f, FILE *output_f);
int OFB_mode_en(uint8_t[10][16], uint8_t *sync,
FILE *input_f, FILE *output_f);
int simple_mode_de(uint8_t [10][16], FILE *input_f, FILE *output_f);
int OFB_mode_de(uint8_t[10][16], uint8_t *sync,
FILE *input_f, FILE *output_f);
```

Заголовочный файл kuzn.h

В заголовочном файле gf.h находятся объявления использующиихся функций из файла gf.c.

```
void gf_tables_init(uint8_t poly);
uint8_t gf_fast_mult(uint8_t a, uint8_t b);.sp 0.5v
```

Заголовочный файл gf.h

Для демонстрационного показа возможностей шифратора был написан заголовочный файл show.h, содержащий макроподстановки, включающиеся при определении макроподсстановки SHOWMODE единицей. В случае, если SHOWMODE равно 1, чтение шифруемого теста происходит в форматированном виде, а на стандартный вывод ошибок подаются промежуточные результаты работы программы, демонстрирующие её внутреннюю работу. В режиме демонстрационного показа в качестве исходного текста стоит использовать только тестовые вектора в шестнадцатеричном виде.

```
#ifndef SHOWMODE
 #define SHOWMODE 1
#endif

#if SHOWMODE == 1
 #define GENERATOR(file, c)(fscanf(file, "%2x", &c)) == EOF
#else
 #define GENERATOR(file, c)(c = fgetc(file))== EOF
#endif

#if SHOWMODE == 1
 #define SHOWPRINT(text, m)\
 do {\
  fprintf (stderr, #text "\n"); \
  int j;\
  for(j = 0; j < 16; j++)\
   fprintf (stderr, "%2X  ", m[j]);\
  fprintf(stderr, "\n\n");\

 } while (0)
#endif

#ifndef SHOWPRINT
 #define SHOWPRINT(arg...)
#endif
```

Заголовочный файл show.h

Стоит так же отметить, что в реализации использвалась 2-я вспомогательная процедура дополнения сообщения.

Заключение

В данной работе были выполнены все поставленные задачи:

- были углублены знания теории написания криптоалгоритмов;
- были подробно изучены и частично описаны ГОСТ 34.12-15 и ГОСТ 34.13-15;
- был программно реализован криптоалгоритм «Кузнечик» в режимах простой замены и гаммирования с обратной связью по шифртексту.

Данная работа является неплохим подспорьем для изучения как российских стандартов на блочное шифрования, так и вышеописанного алгоритма «Кузнечик» в частности как для людей, желающих изучать только теоретический аспект, так и для тех, кто желает выполнить практическуюореализацию. К сожалению, в «Сыктывкарском государственном университете имени Питирима Сорокина» в большинстве случаев предпочтение отдают лишь теоретическому аспекту подобных вопросов.

Список использованных источников

1) W. Diffiee. New directions in cryptography / W. Diffiee, M. Hellman. // IEEE Transactions on Information Theory. — New Jersey, 1976.— Vol 22(6)—pp. 644 - 654.
2) Benvenuto C.J. Galois Field in Cryptography.:May 31, 2012
3) Федеральное агентство по техническому регулированию и метрологии. ГОСТ Р 34.12-2015 Информационная технология КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ Блочные шифры — М.:Стандартинформ,2015. — 25 с.
4) Федеральное агентство по техническому регулированию и метрологии. ГОСТ Р 34.13-2015 Информационная технология КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ Режимы работы блочных шифров — М.:Стандартинформ,2015. — 42 с.

Приложения