**МИНОБРНАУКИ РОССИИ**

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра защиты информации и систем связи

УТВЕРЖДАЮ
Проректор по учебной работе
О.Г. Локтионова
«__» _____ 2015 г.

# ПРОГРАММНАЯ РЕАЛИЗАЦИЯ МОДЕЛИ ПОТОКОВОГО ШИФРАТОРА

Методические указания по выполнению лабораторной работы по дисциплине «Криптографические методы защиты информации» для студентов специальностей 10.05.03, 10.05.02, 10.03.01

Курск 2015

УДК 004.056.55 (076.5)

Составитель: М.А. Ефремов, А.Л. Ханис

Рецензент
Кандидат технических наук, доцент *И.В. Калуцкий*

**Программная реализация модели потокового шифратора:** методические указания по выполнению лабораторной работы по дисциплине «Криптографические методы защиты информации» / Юго-Зап. гос. ун-т; сост.: М.А. Ефремов, А.Л. Ханис. Курск, 2015. 20 с.: ил. 8, табл. 2. Библиогр.: с. 20.

Рассматриваются основные практические и теоретические положения этапов потокового шифрования на примере программной реализации самосинхронизирующегося и аддитивного скремблера. Указывается порядок выполнения лабораторной работы, правила оформления и содержание отчета.

Методические указания соответствуют требованиям программы, утвержденной учебно-методическим объединением по образованию в области информационной безопасности (УМО ИБ).

Предназначены для студентов специальностей 10.05.03, 10.05.02, 10.03.01 дневной формы обучения.

Текст печатается в авторской редакции

Подписано в печать . Формат 60x84 1/16.
Усл.печ. л. . Уч.-изд.л. . Тираж 30 экз. Заказ.
Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

# СОДЕРЖАНИЕ

1. Цель работы ..... 4
2. Задание ..... 4
3. Порядок выполнения работы ..... 4
4. Содержание отчета ..... 4
5. Теоретическая часть ..... 5
5.1. Потоковые алгоритмы ..... 5
5.2. Скремблер ..... 11
6. Практическая часть ..... 16
7. Контрольные вопросы ..... 19
8. Библиографический список ..... 20

# 1. ЦЕЛЬ РАБОТЫ

Целью работы является программная реализация модели потокового шифратора согласно варианту задания.

# 2. ЗАДАНИЕ

Ознакомиться с руководством пользователя и с теоретическим материалом. Программно реализовать модель самосинхронизирующегося и аддитивного скремблера.

# 3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получить задание.
2. Изучить теоретическую часть.
3. Составить блок схему алгоритма.
4. Программно реализовать модель скремблера.

# 4. СОДЕРЖАНИЕ ОТЧЕТА.

1. Титульный лист.
2. Задание согласно варианту.
3. Описание переменных, используемых в программе.
4. Блок – схема алгоритма работы программы.
5. Результаты выполнения работы программы.
6. Выводы о проделанной работе.

# 5. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## 5.1. Потоковые алгоритмы

Потоковые алгоритмы кодируют непрерывный поток данных (например, телефонный разговор), при этом кодирование каждого бита информации осуществляется независимо от остальных битов, что позволяет допускать пропуски в потоке данных. Кодирующие устройства, реализующие потоковые алгоритмы, обычно называют скремблерами. Общий недостаток любого скремблера — наличие фиксированного периода, начиная с которого кодовое слово, дающее возможность осуществлять шифрование, начинает повторяться.

Основная идея потоковых криптосистем заключается в шифровании исходного текста M с помощью криптографического ключа K, длина которого равна длине текста. Каждый бит шифр текста $C_i$ является функцией соответствующих битов исходного текста и ключевого потока:

$$C_i = EK_i(M_i) = M_i \oplus K_i; \quad M_i, K_i, C_i \in \{0,1\} \quad (1)$$

При дешифровании выполняется обратное преобразование $DK_i$:

$$DK_i(C_i) = C_i \oplus K_i = (M_i \oplus K_i) \oplus K_i = M_i \quad (2)$$

Символом «⊕» обозначена операция сложения «ИСКЛЮЧАЮЩЕЕ-ИЛИ». Благодаря линейным свойствам этой операции при шифровании и дешифровании используется одинаковый ключевой поток K. Очевидно, что в этом случае длина K должна быть равна длине передаваемого сообщения. Однако обмен ключами большого размера зачастую невозможен. Поэтому на практике для формирования ключевого потока используют генераторы псевдослучайной последовательности (рис. 1). Начальные параметры I генераторов на стороне отправителя и получателя должны совпадать, они являются секретным ключом алгоритма. Псевдослучайная последовательность каждого генератора обладает определенным периодом, после которого значения повторяются. Поэтому

необходимо выбирать такие генераторы ключа, чтобы этот период превышал длину шифруемой информации.

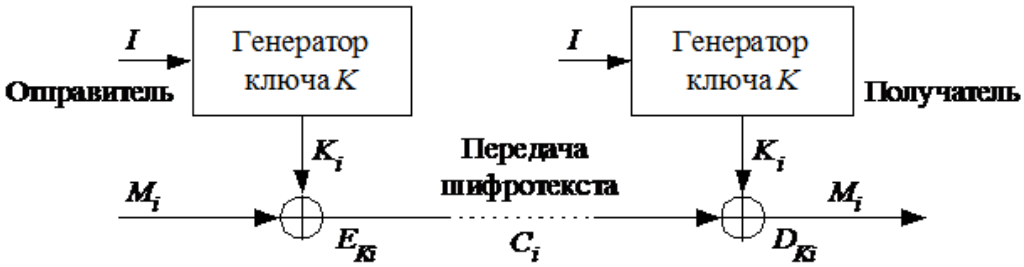



Рисунок 1 - Схема потоковой криптосистемы

Для корректной работы потоковых криптосистем необходимо, чтобы передающая и принимающая сторона имели синхронизированные генераторы ключа K. Искажение отдельных символов не влияет на расшифровку остальных символов шифр текста. Добавление, удаление или дублирование символов шифр текста нарушает синхронизацию ключевой и текстовой последовательностей, и все последующие символы расшифровываются некорректно. Рассмотрим генераторы ключей на основе сдвиговых регистров с линейной обратной связью LFSR (Linear Feedback Shift Register). Они достаточно просто реализуются в программном и аппаратном виде, обладают высокой скоростью генерации и большим периодом ключа. Регистр LFSR состоит из двух частей: сдвигового регистра, выполняющего сдвиг своих разрядов влево на один разряд, и функции обратной связи, вычисляющей вдвигаемое в первый разряд значение. В обобщенном виде структура LFSR представлена на рисунке 2.

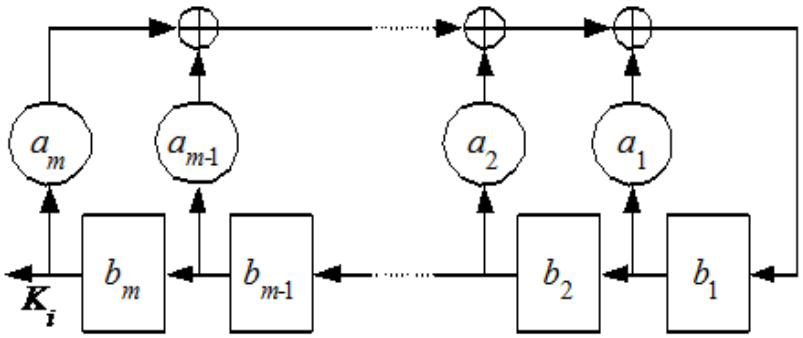



Рисунок 2 - Обобщенная схема LFSR

Структуру конкретного регистра LFSR принято задавать с помощью характеристического (задающего) многочлена:

$$P(x) = a_m x^m + a_{m-1} x^{m-1} + \ldots + a_2 x^2 + a_1 x + 1,\ a_k \in \{0,1\}, k = 1,\ldots,\mathrm{m} \quad (3)$$

Степень многочлена $m$ задает длину сдвигового регистра. Ненулевые коэффициенты $a_k$ определяют разряды регистра, которые будут участвовать в формировании вдвигаемого в первый разряд значения. Через $b_k, (b_k \in \{0,1\})$ обозначены текущие значения разрядов LFSR ($k$= 1, …,$m$). Новые значения разрядов $b_k$, $(b_k \in \{0,1\})$ вычисляются по следующему закону:

$$b_k = \begin{cases} \sum_{j=1}^{m} \oplus a_j b_j, & k=1; \\ 1 & \\ b_{k-1}, & k = 2,\ldots,\mathrm{m} \end{cases} \quad (4)$$

Таким образом, новое значение для всех разрядов, кроме первого, берется из ближайшего младшего разряда. Символом $\sum\oplus$ обозначена многоместная операция сложения «ИСКЛЮЧАЮЩЕЕ-ИЛИ». Она используется для сложения значений из разрядов, которые отмечены ненулевыми коэффициентами $a_k$ характеристического многочлена. Полученная сумма вдвигается в первый разряд LFSR. Очередной бит ключа $K_i$ для потоковой криптосистемы равен значению старшего разряда LFSR $b_m$.

Пример 1. Построим сдвиговый регистр с линейными обратными связями, задаваемый характеристическим многочленом $P(x) = x^4 + x + 1$.

Схема регистра приведена на рисунке 3. Если начальное состояние регистра $I$ =1111, то последовательность генерируемых состояний имеет вид

$$1111 \to 1110 \to 1101 \to 1010 \to 0101 \to 1011 \to 0110 \to 1100 \to$$
$$1001 \to 0010 \to 0100 \to 1000 \to 0001 \to 0011 \to 0111 \to 1111$$

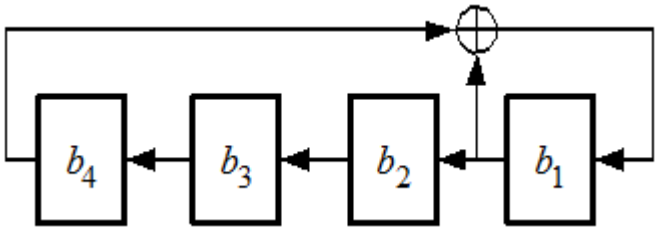



Рисунок 3 - LFSR на основе многочлена

Последнее состояние совпадает с начальным, после этого указанная последовательность повторяется. Последовательность сгенерированных бит ключа: 1111010110010001.

Свойства генерируемой последовательности определяются постоянными коэффициентами характеристического многочлена $a_i$. При их соответствующем выборе генерируемая последовательность K будет иметь максимально возможный период, равный – 1. Последовательность максимально возможного для данного генератора периода называется M-последовательностью. Основная задача синтеза генератора на основе LFSR – нахождение характеристического многочлена, позволяющего сформировать M-последовательность. Для того чтобы конкретный LFSR имел максимальный период, соответствующий многочлен должен быть примитивным. В общем случае не существует простого способа нахождения примитивных многочленов данной степени, проще выбирать многочлен случайным образом и проверять, является ли он примитивным. Эта задача аналогична задаче проверки на простоту случайно выбранного целого числа и решается с помощью вычислительной техники. Таблица 1 содержит некоторые примитивные многочлены.

Таблица 1 – Примитивные многочлены

| m | P(x) | m | P(x) |
|---|---|---|---|
| 23 | $X^{23}+X^5+1$ | 29 | $X^{29}+X^2+1$ |
| 24 | $X^{24}+X^4+X^3+X+1$ | 30 | $X^{30}+X^{16}+X^{15}+X+1$ |
| 25 | $X^{25}+X^3+1$ | 31 | $X^{31}+X^3+1$ |
| 26 | $X^{26}+X^8+X^7+X+1$ | 32 | $X^{32}+X^{28}+X^{27}+X+1$ |
| 27 | $X^{27}+X^8+X^7+X+1$ | 33 | $X^{33}+X^{13}+1$ |
| 28 | $X^{28}+X^3+1$ | 34 | $X^{34}+X^{15}+X^{14}+X+1$ |

Использование LFSR для создания потоковых криптосистем предполагает наличие уязвимости, связанной с линейным характером генерируемой последовательности. Обладая парой «исходный текст – шифр текст» длиной всего 2m бит, взломщик может восстановить характеристический многочлен и дешифровать все сообщение. Для защиты от этой атаки следует

увеличивать размер используемого LFSR или использовать более сложные схемы генерации ключа. Рассмотрим, для примера, генератор Геффе (Geffe), представленный на рисунке 4.

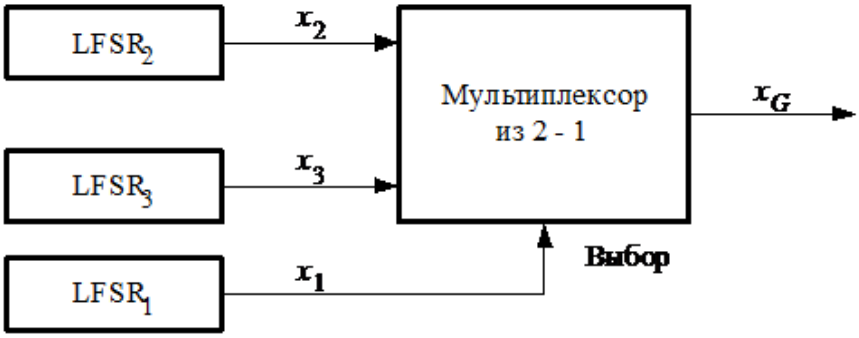



Рисунок 4 – Генератор Геффе

В нем используются три регистра с линейной обратной связью, объединённые нелинейным образом. Два регистра LFSR являются входами мультиплексора, а третий – управляет его выходом. Через $x_1$, $x_2$ и $x_3$ обозначены выходы трёх LFSR, выход генератора Геффе $x_G$ описывается так.

$$x_G = (x_1 \; and \; x_2) \, or \, (not \; x_1 \; and \; x_3) \quad (5)$$

Период данного генератора равен $(2^{m_1}-1)(2^{m_2}-1)(2^{m_3}-1)$, где $m_1$, $m_2$ и $m_3$ – длины первого, второго и третьего LFSR соответственно.

Благодаря простоте реализации, высокой скорости работы и сравнительно высокой криптостойкости потоковые шифраторы получили широкое распространение для шифрования информации средней степени секретности. Например, алгоритм A5, построенный на основе трех LFSR с прореженными обратными связями, входит в состав стандарта мобильной связи GSM.

Возможны и другие способы генерации ключевой последовательности. Например, генератор ключевого потока K в алгоритме RC4 работает на основе подстановочной таблицы S (S-бокса) из 256 символов. На первом шаге S-бокс заполняется линейно: S[0] = 0, S[1] = 1, …, S[255] = 255. Затем начальное значение S-бокса меняется на основе пользовательского секретного ключа U по следующему алгоритму:

$$
\begin{aligned}
&j := 0;\\
&for\, i := 0\, to\, 255\\
&j := \left(j + S[i] + U\left[i \bmod length(U)\right]\right) mod\ 256;\\
&swap(S[i], S[j]);
\end{aligned}
\qquad (6)
$$

Эта подстановка является функцией от ключа изменяемой длины. Процедура swap меняет местами значения таблицы S с заданными индексами. Полученное значение S-бокса используется для побайтной генерации ключевого потока K. Генератор потока имеет два счетчика i и j, инициализируемых нулевым значением. На каждом шаге генерации выполняются следующие операции:

$$
\begin{aligned}
&i := (i + 1) mod\ 256\\
&j := \left(j + S[i]\right) mod\ 256\\
&swap(S[i], S[j])\\
&K := S\left[(S[i] + S[j]) mod\ 256\right]
\end{aligned}
\qquad (7)
$$

Байт K складывается операцией «ИСКЛЮЧАЮЩЕЕ-ИЛИ» с байтом открытого текста для получения байта шифротекста либо с байтом шифротекста для получения байта исходного текста. Шифрование происходит весьма быстро – примерно в 10 раз быстрее DES-алгоритма. Типичная реализация выполняет 19 машинных команд на каждый байт текста. S-бокс медленно изменяется в процессе работы: параметр i обеспечивает изменение каждого элемента, а j отвечает за то, чтобы эти элементы изменялись случайным образом. Шифр обладает иммунитетом к методам линейного и дифференциального криптоанализа и до сих пор в нем не обнаружены короткие циклы. Алгоритм RC4, в отличие от алгоритмов на основе сдвиговых регистров LSFR, больше ориентирован на программную реализацию, поскольку работает не с битами, а с целыми байтами исходного текста. Благодаря своей скорости и защищенности, он нашел широкое применение в криптографических системах. Например, он является частью протокола безопасного обмена информацией SSL.

## 5.2. Скремблер

Скремблирование — это обратимое преобразование цифрового потока без изменения скорости передачи с целью получения свойств, близких к свойствам случайной последовательности. Исходное сообщение можно восстановить, применив обратный алгоритм. Применительно к телекоммуникационным системам скремблирование повышает надежность синхронизации устройств, подключенных к линии связи, и уменьшает уровень помех, излучаемых на соседние линии многожильного кабеля. Есть и иная область применения скремблеров — защита передаваемой информации от несанкционированного доступа. Для синхронной передачи двоичный сигнал должен удовлетворять двум основным требованиям:

- частота смены символов (1,0) должна обеспечивать надежное выделение тактовой частоты непосредственно из принимаемого сигнала;
- спектральная плотность мощности передаваемого сигнала должна быть, по возможности, постоянной и сосредоточенной в заданной области частот с целью снижения взаимного влияния каналов.

Одним из способов обработки двоичных посылок, удовлетворяющим данным требованиям является скремблирование (scramble – перемешивание). После скремблирования появление «1» и «0» в выходной последовательности примерно равновероятно. Скремблирование также может использоваться для определенной защиты передаваемой информации, а также для идентификации абонентов. Скремблирование широко применяется во многих видах систем связи для улучшения статистических свойств сигнала. Обычно скремблирование осуществляется на последнем этапе цифровой обработки непосредственно перед модуляцией (рисунок 5).

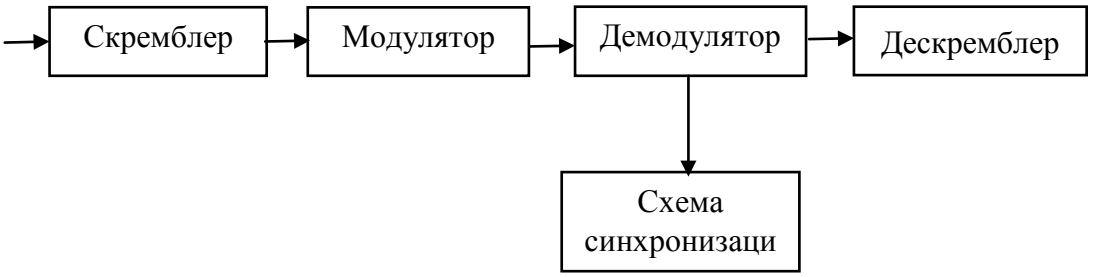



Рисунок 5 - Схема включения скремблера и дескремблера в канал связи

Скремблирование производится на передающей стороне с помощью устройства – скремблера, реализующего логическую операцию суммирования по модулю 2 исходного и преобразующего псевдослучайного двоичного сигнала. На приемной стороне осуществляется обратная операция – восстановление устройством, называемым дескремблером. Дескремблер выделяет из принятой последовательности исходную последовательность.

Основной частью скремблера является генератор псевдослучайной последовательности (ПСП) в виде линейного n-каскадного регистра с обратными связями, формирующий последовательность максимальной длины $2^n - 1$.

Различают два основных типа скремблеров и дескремблеров - самосинхронизирующиеся (СС) и с установкой (аддитивные). В литературе также можно встретить другие названия – скремблеры с неизолированным и изолированным от линии связи генераторами псевдослучайных последовательностей.

Особенностью самосинхронизирующегося скремблера (СС скремблера) (Рисунок 6) является то, что он управляется скремблированной последовательностью, т.е. той, которая передается в канал. Поэтому при данном виде скремблирования не требуется специальной установки состояний скремблера и дескремблера; скремблированная последовательность записывается в регистры сдвига скремблера и дескремблера, устанавливая их в идентичное состояние. При потере синхронизма между скремблером и дескремблером время восстановления синхронизма не превышает числа тактов, равного числу ячеек регистра скремблера.

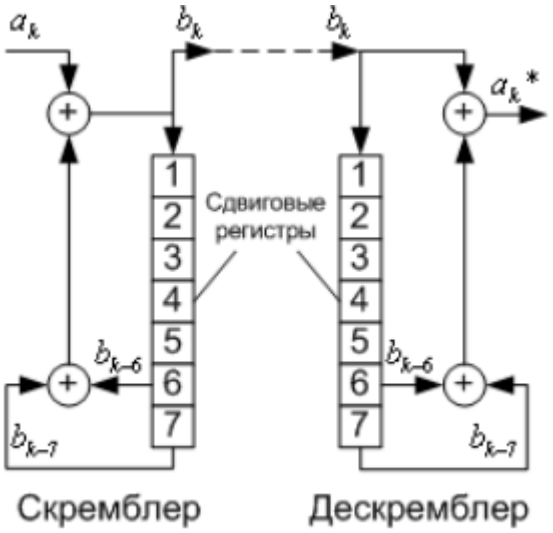



Рисунок 6 - Самосинхронизирующиеся скремблер и дескремблер

На приемной стороне выделение исходной последовательности происходит путем сложения по модулю 2 принятой скремблированной последовательности с последовательностью на выходе сдвигового регистра. Например, для схемы Рис. 6 входная последовательность с помощью скремблера в соответствии с соотношением $b_k = a_k xor\left(b_{k-6} xor b_{k-7}\right)$ преобразуется в посылаемую двоичную последовательность bk. В приемнике из этой последовательности таким же регистром сдвига, как на приеме, формируется последовательность $a*_k = b_k xor\left(b_{k-6} xor b_{k-7}\right)$. Эта последовательность на выходе дескремблера идентична первоначальной последовательности $a_k$.

Как следует из принципа действия схемы, при одной ошибке в последовательности $b_k$ ошибочными получаются также последующие шестой и седьмой символы (в данном примере). В общем случае влияние ошибочно принятого бита будет сказываться $\alpha$ раз, где $\alpha$ - число обратных связей. Таким образом, СС скремблер-дескремблер обладает свойством размножения ошибок. Данный недостаток СС скремблера-дескремблера ограничивает число обратных связей в регистре сдвига; практически это число не превышает $\alpha$ =2.

Второй недостаток СС скремблера связан с возможностью появления на его выходе при определенных условиях так называемых «критических ситуаций», когда выходная последовательность приобретает периодический характер с периодом, меньшим длины ПСП. Чтобы предотвратить это, в скремблере и дескремблере предусматриваются специальные дополнительные схемы контроля, которые выявляют наличие периодичности элементов на входе и нарушают ее.

Недостатки, присущие СС скремблеру-дескремблеру, практически отсутствуют при аддитивном скремблировании (Рисунок 7), однако, этот тип скремблеров-дескремблеров требует предварительной идентичной установки состояний регистров скремблера и дескремблера. В скремблере с установкой (АД-скремблере), как и в СС скремблере, производится суммирование входного сигнала и ПСП, но результирующий сигнал не поступает на вход регистра. В дескремблере скремблированный сигнал также не проходит через регистр сдвига, поэтому размножения ошибок не происходит

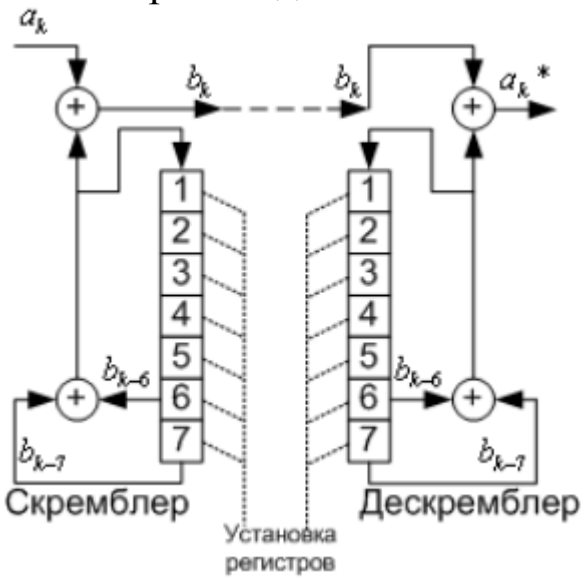



Рисунок 7 - Аддитивные скремблер и дескремблер

Суммируемые в скремблере последовательности независимы, поэтому их период всегда равен наименьшему общему краткому величин периодов входной последовательности и ПСП и критическое состояние отсутствует. Отсутствие эффекта размножения ошибок и необходимости в специальной логике защиты от нежелательных ситуаций делают способ аддитивного

скремблирования предпочтительнее, если не учитывать затрат на решение задачи фазирования скремблера и дескремблера. В качестве сигнала установки в ЦСП используют сигнал цикловой синхронизации.

Скремблирование так же влияет на энергетический спектр двоичного сигнала (Рисунок 8).

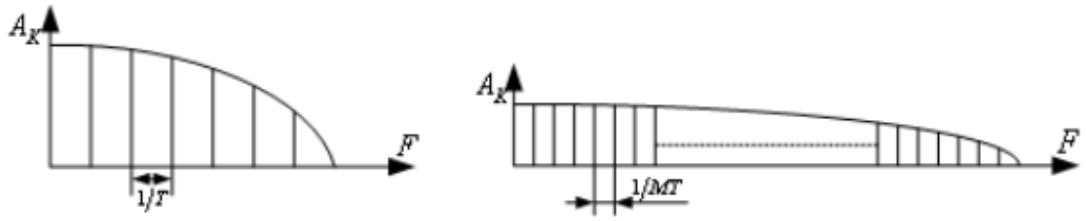



Рисунок 8 - Спектр сигнала до скремблирования (слева), после скремблирования (справа)

На рисунке 8 слева изображен пример энергетического спектра для периодического сигнала с периодом Т, содержащим 6 двоичных элементов с длительностью $T_0$. После скремблирования ПСП с $M = 2^n - 1$ элементами спектр существенно «обогащается» (рисунок 8 справа). При этом число составляющих спектра увеличилось в М раз, причем, уровень каждой составляющей уменьшается в такое же число раз.

# 6. ПРАКТИЧЕСКАЯ ЧАСТЬ

Выбрать вариант задания из таблицы 2 согласно номеру варианта. Написать программу на языке высокого уровня, выполняющую:

1. Ввод текста с клавиатуры.
2. Создать функцию, реализующую работу скремблера согласно варианту. Зашифровать введенный текст. В качестве входных параметров использовать:
    a. Последовательность 0 и 1, которая должна быть скремблирована.
    b. Исходное значение сдвигового регистра (если требуется).
3. Вывести значения регистров на каждом этапе шифрования.
4. Создать функцию, реализующую работу дескремблера согласно варианту. В качестве входных параметров использовать:
    a. Скремблированную последовательность.
    b. Исходное значение сдвигового регистра (если требуется).
5. Печать раскодированной строки.

Таблица 2 – Варианты заданий

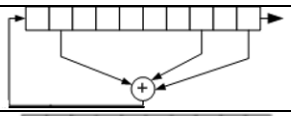
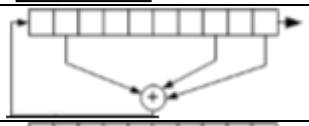
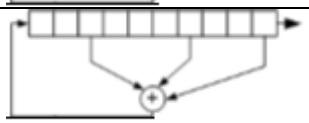
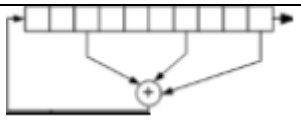
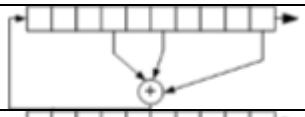
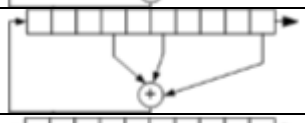
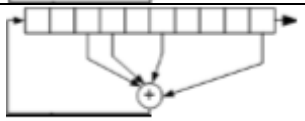
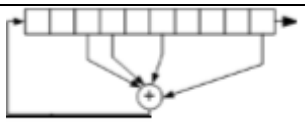
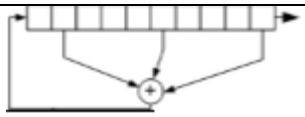
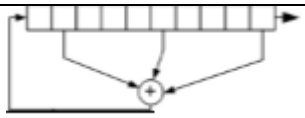
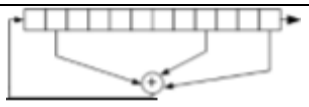
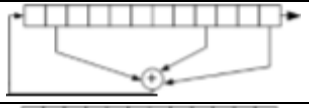
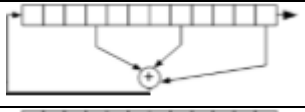
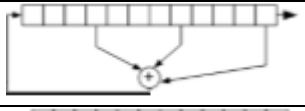
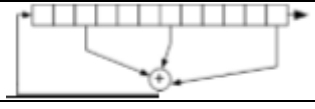
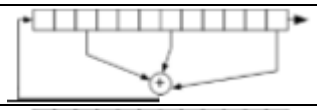
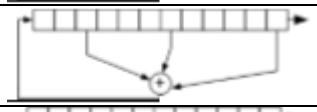
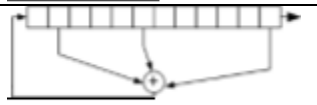
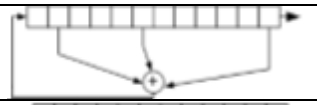
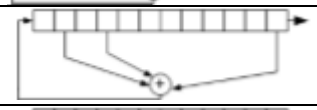
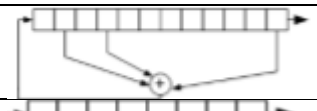
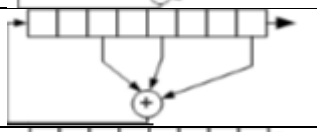
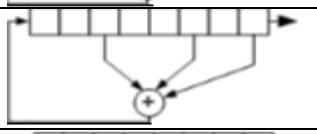
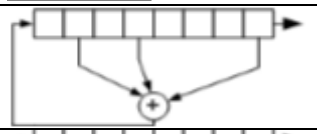
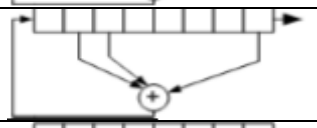
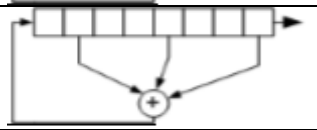
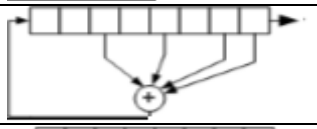
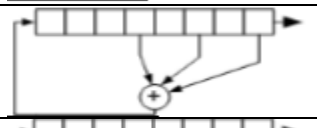
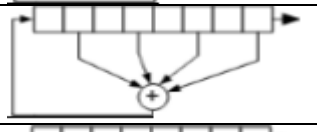
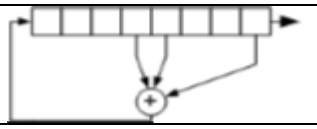
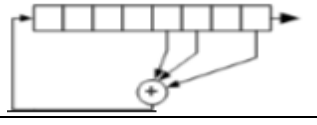
| № | Тип скремблера/ дескремблера | | Сдвиговой регистр | Исходное значение |
|---|---|---|---|---|
| 1 | Самосинхрони-зирующийся | 10 |  | 1010000010 |
| 2 | Аддитивный | 10 |  | 0010010110 |
| 3 | Самосинхрони-зирующийся | 10 |  | 1001000101 |
| 4 | Аддитивный | 10 |  | 0000100100 |
| 5 | Самосинхрони-зирующийся | 10 |  | 1000101001 |
| 6 | Аддитивный | 10 |  | 1111111111 |
| 7 | Самосинхрони-зирующийся | 10 |  | 1001101000 |
| 8 | Аддитивный | 10 |  | 1111111111 |
| 9 | Самосинхрони-зирующийся | 10 |  | 1010001000 |
| 10 | Аддитивный | 10 |  | 1111111111 |
| 11 | Самосинхрони-зирующийся | 12 |  | 101000000100 |
| 12 | Аддитивный | 12 |  | 000101001101 |
| 13 | Самосинхрони-зирующийся | 12 |  | 100010001001 |
| 14 | Аддитивный | 12 |  | 110101001001 |
| 15 | Самосинхрони-зирующийся | 12 |  | 100100010000 |

Таблица 2 - Продолжение

| | | | | |
|---|---|---|---|---|
| 15 | Самосинхрони-зирующийся | 12 |  | 100100010000 |
| 16 | Аддитивный | 12 |  | 110001010001 |
| 17 | Самосинхрони-зирующийся | 12 |  | 101000100000 |
| 18 | Аддитивный | 12 |  | 001011010100 |
| 19 | Самосинхрони-зирующийся | 12 |  | 101010100000 |
| 20 | Аддитивный | 12 |  | 100010101101 |
| 21 | Самосинхрони-зирующийся | 8 |  | 10010100 |
| 22 | Аддитивный | 8 |  | 00110010 |
| 23 | Самосинхрони-зирующийся | 8 |  | 10101000 |
| 24 | Аддитивный | 8 |  | 00101010 |
| 25 | Самосинхрони-зирующийся | 8 |  | 10100101 |
| 26 | Аддитивный | 8 |  | 00100011 |
| 27 | Самосинхрони-зирующийся | 8 |  | 10001010 |
| 28 | Аддитивный | 8 |  | 10010110 |
| 29 | Самосинхрони-зирующийся | 8 |  | 10001100 |
| 30 | Аддитивный | 8 |  | 11100111 |

# 7. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие бывают алгоритмы шифрования?
2. Что такое потоковый шифр?
3. Что такое скремблер?
4. Что такое дескремблер?
5. Для каких целей используют скремблеры и дескремблеры?
6. Какие типы скремблеров и дескремблеров вам известны?
7. Какие преимущества и недостатки самосинхронизирующихся скремблеров и дескремблеров вам известны?
8. Какие преимущества и недостатки аддитивных скремблеров и дескремблеров вам известны?

# 8. БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. В. И. Нечаев. Элементы криптографии. Основы теории защиты информации. – М.: Высшая школа. 1999

2. Венбо Мао. Современная криптография. Теория и практика. – М.:Вильмас. 2005.

3. М. А. Иванов. Криптография. Криптографические методы защиты информации в компьютерных системах и сетях. – М.: КУДИЦ-Образ. 2001.

4. Ю.В. ВЕТРОВ С.Б. МАКАРОВ. Криптографические методы защиты информации в телекоммуникационных системах. – СПб.: Изд - во Политехн. ун-та, 2011. — 174 с.

5. Иванов М.А., Чугунков И.В. Криптографические методы защиты информации в компьютерных системах и сетях. Учебное пособие - Москва: МИФИ, 2012.- 400 с.