Практическая работа N 9

Поточное симметричное шифрование

Цель работы: изучение структуры и основных принципов работы современных алгоритмов поточного симметричного шифрования, приобретение навыков программной реализации поточных симметричных шифров.

Основные сведения

Поточные шифры характерны тем, что шифруют информацию по одному биту за такт шифрования. Учитывая, что среди операций с битами существуют только две обратимые – сумма по модулю 2 и логическое отрицание, то выбор принципа шифрования очевиден – биты открытого текста должны складываться с битами ключевой последовательности с помощью операции $\oplus$:

$c_i = m_i \oplus k_i.$

Расшифрование происходит аналогичным образом:

$m_i = c_i \oplus k_i.$

Учитывая свойства операции сложения по модулю 2, можно отметить, что выполняется:

$k_i = c_i \oplus m_i,$

поэтому криптостойкость поточных шифров полностью зависит от качества генератора потока ключей. Очевидно, что если поток ключей будет включать в себя только двоичные нули, то шифротекст будет представлять собой точную копию открытого текста. Поток ключей поточных шифров принято обозначать греческой буквой γ (гамма), вследствие чего подобные шифры получили название шифров гаммирования. Рассмотрим основные методы формирования γ в современной потоковой криптографии.

Очень популярны для решения этой цели ***регистры сдвига с обратной связью***. Регистр представляет собой (рис.1) последовательность бит, которая на каждом такте шифрования сдвигается вправо на 1 разряд, при этом *выход* из крайнего *правого* бита является выходом генератора, а на *вход* крайнего *левого* бита подается *значение*, вычисляемое как некоторая ***функция*** от отдельных битов регистра. Ключ шифрования поточного шифра заносится в регистр перед началом генерации гаммы.

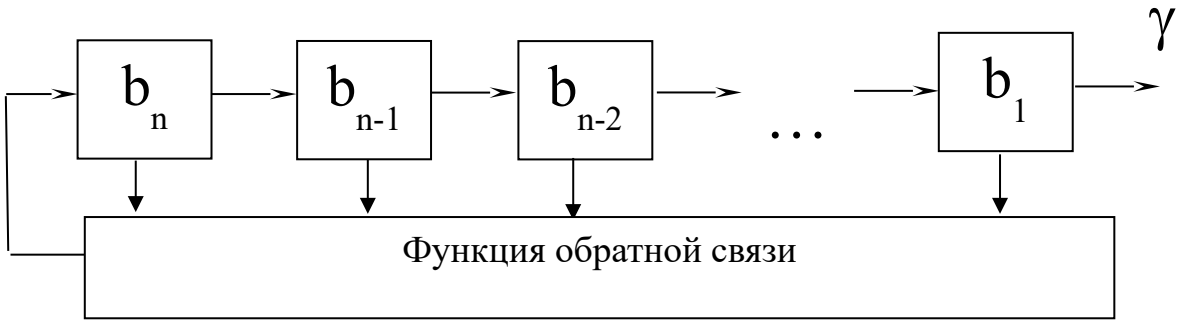


Рис.1. Регистр сдвига с обратной связью

Самым простым способом формирования обратной связи является с*уммирование по модулю 2 отдельных разрядов регистра*. На рис.2 изображен регистр сдвига с линейной обратной связью,

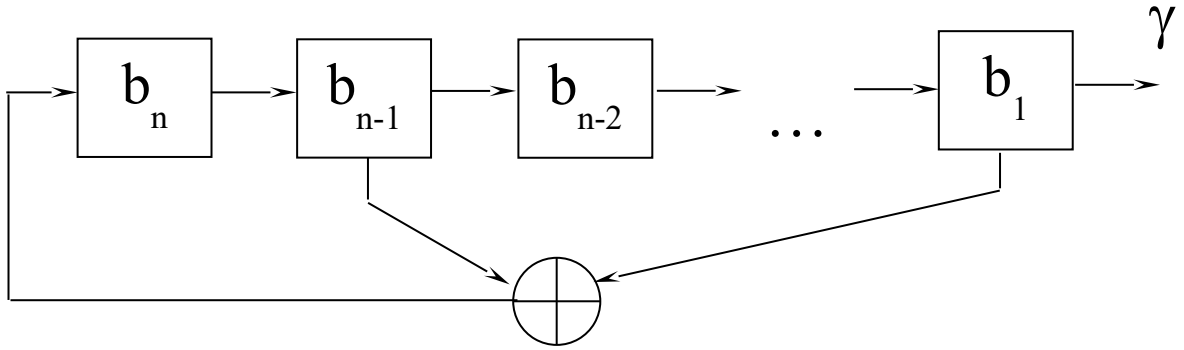


Рис.2. Линейный регистр сдвига

Рассмотрим работу линейного регистра сдвига (ЛРС) на примере трехразрядного регистра, структура которого приведена на рис.3.

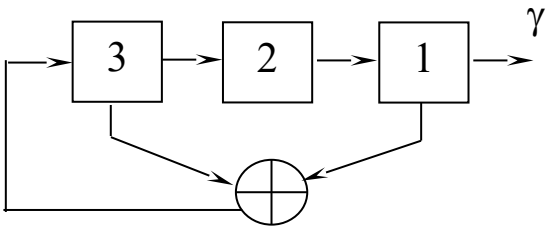


Рис3 Линейный регистр сдвига на три разряда

Занесем в регистр начальное значение 010 и посмотрим, какие значение получим на выходе гаммы (табл.1).

Таблица 1. Результат работы генератора гаммы на основе ЛРС

Номер такта	Значения битов ЛРС			Бит гаммы
	3	2	1	
нач.сост	**0**	**1**	**0**	-
1	0	0	1	0
2	1	0	0	1
3	1	1	0	0
4	1	1	1	0
5	0	1	1	1
6	1	0	1	1
7	0	1	0	1
8	0	0	1	0

Из таблицы видно, что состояние ЛРС повторяется через 7 тактов (начальное состояние ЛРС совпадает с его состоянием на 7-м такте). Повтор состояния ЛРС означает, что и *гамма* будет периодически повторяться. Повторение гаммы снижает криптостойкость поточных шифров, позволяя криптоаналитику проводить анализ шифротекстов, полученных кодированием на одной и той же гамме. Поэтому при проектировании структуры ЛРС встает проблема достижения максимального периода повтора ЛРС. Для ЛРС длиной n бит максимальный период составляет 2^n-1 тактов (состояние, когда все биты равны нулю, недопустимо, поскольку ЛРС любой структуры не выходит из этого состояния, зацикливаясь в нем). Построение ЛРС оптимальной структуры с точки зрения периода повторения гаммы имеет четкую математическую основу в виде теории неприводимых полиномов.

Структура ЛРС описывается многочленом вида:

$$b_n*x^n+b_{n-1}*x^{n-1}+b_{n-2}*x^{n-2}+\ldots+b_2*x^2+b_1*x+1, \quad (1)$$

где b_i=0, если i-й бит не участвует в обратной связи, и b_i=1, если участвует. ЛРС будет иметь максимально возможный период повторения гаммы, если описывающий его многочлен не раскладывается на произведение многочленов меньшей степени, то есть является примитивным по модулю 2. В контексте (1) структуру ЛРС принято коротко обозначать записью вида (43, 21, 5, 4, 1, 0), что в данном конкретном случае означает построение обратной линейной связи на сорок третьем, двадцать первом, пятом, четвертом и первом разрядах ЛРС. Отметим, что часто также используется понятие «регистр сдвига с линейной обратной связью».

Основной проблемой ЛРС является их нестойкость к атаке на основе известного открытого текста. Даже если неизвестна внутренняя структура ЛРС, криптоаналитик с помощью алгоритма Берлекэмпа-Мэсси по известным $2N$ битам открытого текста и соответствующего шифротекста имеет возможность построить ЛРС, порождающую подобную последовательность (проблема линейной сложности ЛРС). Поэтому современные поточные шифры строятся на основе нелинейных схемах объединения ЛРС, которые добавляют в структуру нелинейные элементы: логическое сложение и логическое умножение. Наиболее популярными классами нелинейных схем подключения на сегодня являются *фильтрующие*, *комбинирующие* и *динамические*

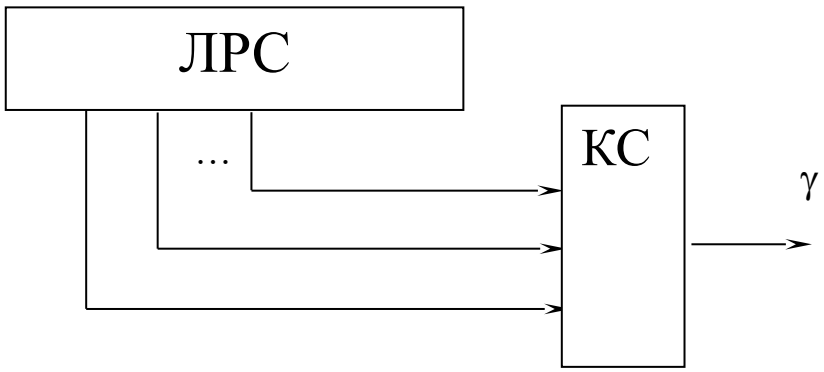


Рис.4. Поточный шифр на основе фильтрующего НРС

поточные шифры [4].

Фильтрующие схемы строятся с использованием дополнительной комбинационной схемы – фильтра – на выходах некоторых бит ЛРС (рис.4). Выход комбинационной схемы и является гаммой.

Комбинирующие схемы также используют комбинационную схему с нелинейными преобразованиями бит, но на вход этой комбинационной схемы подаются выходы нескольких ЛРС (рис.5).

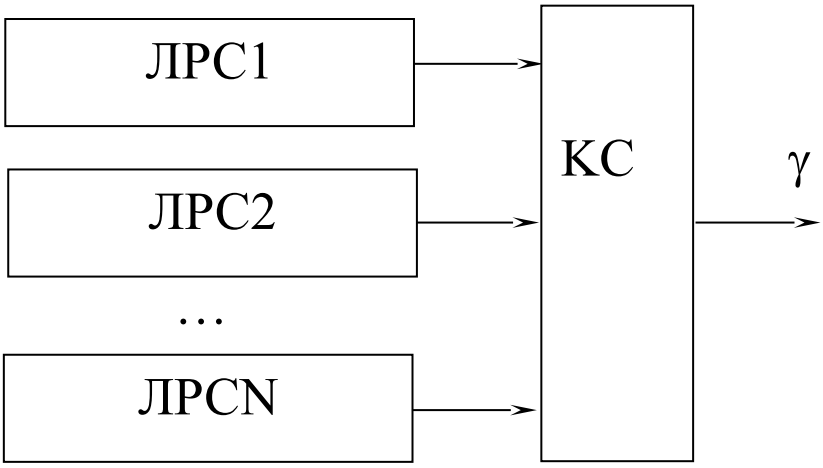


Рис.5. Поточный шифр на основе комбинирующей схемы

Динамические схемы объединения ЛРС предполагают отношения «главный-подчиненный» между отдельными регистрами. Например, на схеме рис. 6. зависимости от выхода управляющего ЛРС на общий выход гаммы подается либо выход первого, либо второго ЛРС.

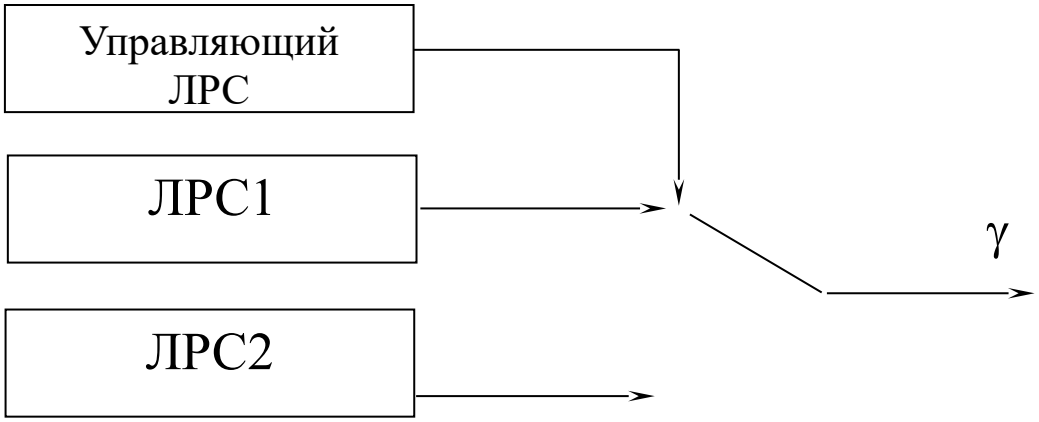


Рис.6. Поточный шифр на основе динамического НРС

Существуют также схемы динамического подключения ЛРС с использованием управляемого тактирования, когда сдвиг управляемого ЛРС зависит от состояния некоторого бита управляющего ЛРС.

В качестве примера ***поточного*** шифра, построенного на основе регистров сдвига, можно привести алгоритм **A5**, используемый для кодирования в стандарте GSM. A5 включает 3 ЛРС длиной 19, 22 и 23 бита на выход гаммы подается сумма по модулю 2 выходов всех регистров. Используется динамическая схема (рис.7), когда каждый регистр тактируется в зависимости от состояния средних разрядов всех трех регистров сдвига.

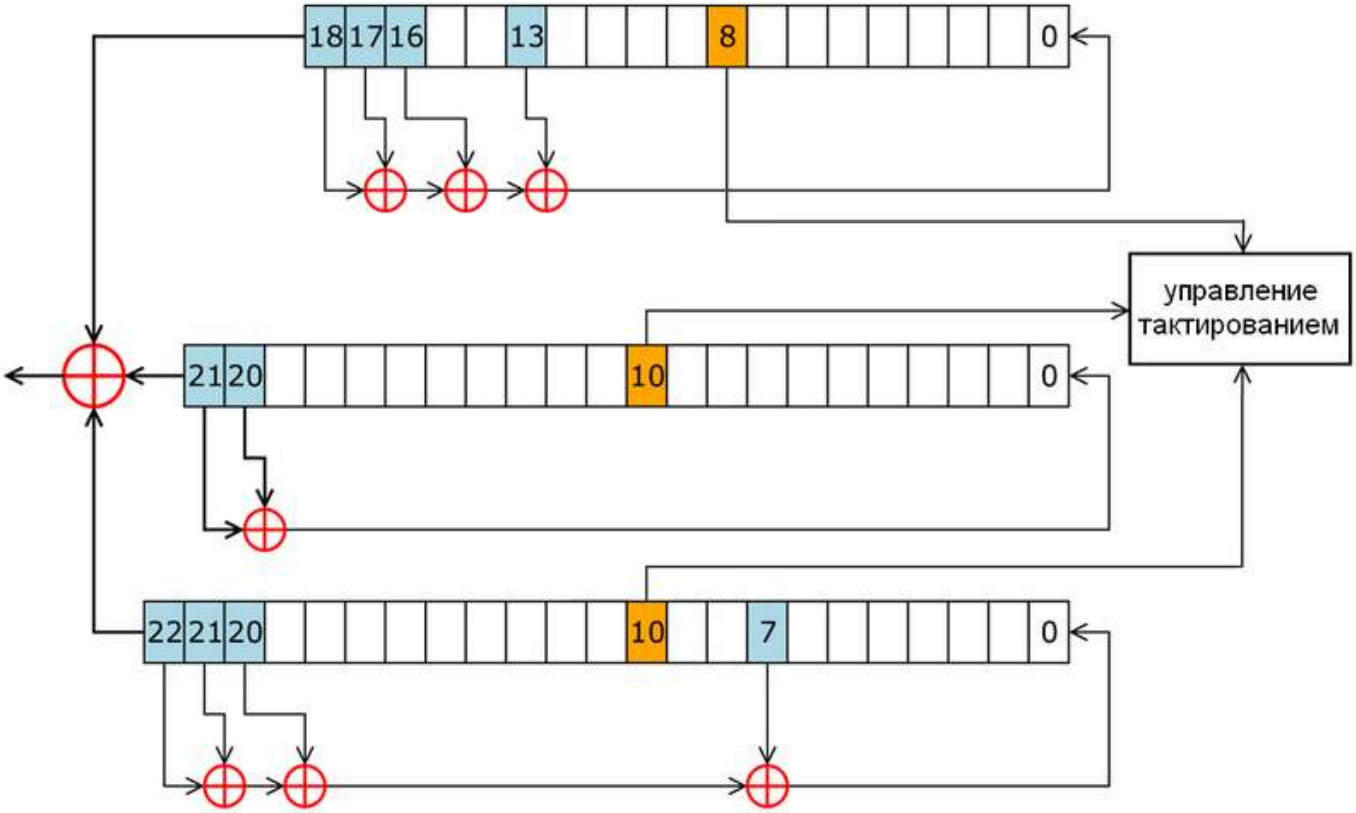


Рис. 7. Структура алгоритма шифрования A5

Каждый регистр содержит биты синхронизации: 8 (в R1), 10 (в R2), 10 (в R3). Для управления тактированием вычисляется функция F = x&y|x&z|y&z, где x, y и z — биты синхронизации R1, R2 и R3 соответственно. Сдвигу подвергаются лишь те регистры, у которых бит синхронизации равен F.

При построении нелинейных схем на основе нескольких ЛРС необходимо помнить о корреляционной зависимости между выходом общего генератора гаммы и выходом конкретного ЛРС. Она может выражаться в том, что выход всей схемы и выход некоторого ЛРС, входящего в ее состав, совпадают существенно более чем в 50 %

случаев. Если из-за слабости предложенной схемы такая зависимость существует, то анализ всего генератора гаммы можно свести к анализу лишь одного ЛРС, что, как уже упоминалось выше, несложно.

Итак, констатируем...

Регистр сдвига с линейной обратной связью (РСЛОС) – упорядоченный набор битов, у которого значение входного (вдвигаемого слева, старшего) бита равно линейной булевой функции от значений остальных битов регистра до сдвига. Теорию последовательности регистров сдвига разработал в 1965 г. главный криптограф норвежского правительства Эрнст Селмер.

Длиной регистра называется количества битов в нем. В качестве булевой функции для РСЛОС чаще всего используют сложение по модулю 2. Такие РСЛОС обычно записывают в виде *многочлена* (полинома) или упорядоченной последовательности. Например, запись $x^8 + x^4 + x^3 + x^2 + 1$ или короткая запись РСЛОС(8, 4, 3, 2, 0) означает, что длина регистра 8 битов, а входной бит рассчитывается по формуле $b_7 = b_4 \oplus b_3 \oplus b_2 \oplus b_0$ (для битов нумерация начинается справа и с нулевой позиции). Выходной (выдвигаемый справа, младший) бит будет являться частью генерируемой гаммы. Расчет требуемой гаммы осуществляется в цикле, на каждой итерации которого выполняются следующие операции.

1. Добавление выходного бита b_0 к гамме.
2. Расчет входного бита b_{n-1} по формуле.
3. Сдвиг битов регистра вправо на одну позицию.
4. Занесение рассчитанного входного бита b_{n-1} в позицию n-1.

В следующей таблице приведен пример генерации гаммы для регистра, инициализированного значением 10001100_2.

Таблица. Пример генерации гаммы для РСЛОС $x^8 + x^4 + x^3 + x^2 + 1$

№ п/п	Исходное состояние регистра	Бит гаммы, b_0	Входной бит, $b_7 = b_4 \oplus b_3 \oplus b_2 \oplus b_0$	Сдвиг регистра вправо на одну позицию	Занесение входного бита b_7 в регистр
0	1 0 0 0 1 1 0 0	0	$0 \oplus 1 \oplus 1 \oplus 0 = 0$	1 0 0 0 1 1 0	0 1 0 0 0 1 1 0
1	0 1 0 0 0 1 1 0	0	$0 \oplus 0 \oplus 1 \oplus 0 = 1$	0 1 0 0 0 1 1	1 0 1 0 0 0 1 1
2	1 0 1 0 0 0 1 1	1	$0 \oplus 0 \oplus 0 \oplus 1 = 1$	1 0 1 0 0 0 1	1 1 0 1 0 0 0 1
3	1 1 0 1 0 0 0 1	1	$1 \oplus 0 \oplus 0 \oplus 1 = 0$	1 1 0 1 0 0 0	0 1 1 0 1 0 0 0
4	0 1 1 0 1 0 0 0	0	$0 \oplus 1 \oplus 0 \oplus 0 = 1$	0 1 1 0 1 0 0	1 0 1 1 0 1 0 0
...	...	...	...	...	...
Гамма		00110...			

Для регистра длиной n максимальное количество состояний составляет 2^n–1. Это число равно 2^n–1, а не 2^n, поскольку заполнение РСЛОС нулями влечет вывод регистром бесконечной последовательности нулей, что совершенно бесполезно. Таким образом, максимально возможная длина периода гаммы в битах составляет $(2^n–1)n$. Для получения такого максимального периода многочлен должен быть примитивным по модулю 2. **Примитивный многочлен степени n** – неприводимый

многочлен, который является делителем $x^{2^{n-1}}+1$, но не является делителем x^d+1 для всех d, являющихся делителями 2^n-1. Чем больше битов регистра используются для расчета входного бита, тем лучше (повышается стойкость). Многочлены с большим количеством членов называют **плотными**, с малым – **разреженными**.

РСЛОС обладают высокой скоростью генерации чисел, хорошими статистическими свойствами, а также возможностью простой реализации на аппаратном уровне. Исследователями предложено несколько десятков различных вариантов реализации генераторов на базе РСЛОС, в т.ч. с применением комбинации нескольких РСЛОС одновременно и разных функций расчета входного или выходного бита (регистры сдвига с нелинейной обратной связью, обобщенной обратной связью, обратной связью по переносу и т.д.).

Еще одной разновидностью сдвиговых регистров, использующихся для генерации потока ключей в потоковых шифрах, являются ***сдвиговые регистры с обратной связью по переносу*** (РОСП). Структура подобного регистра приведена на рис. 8.

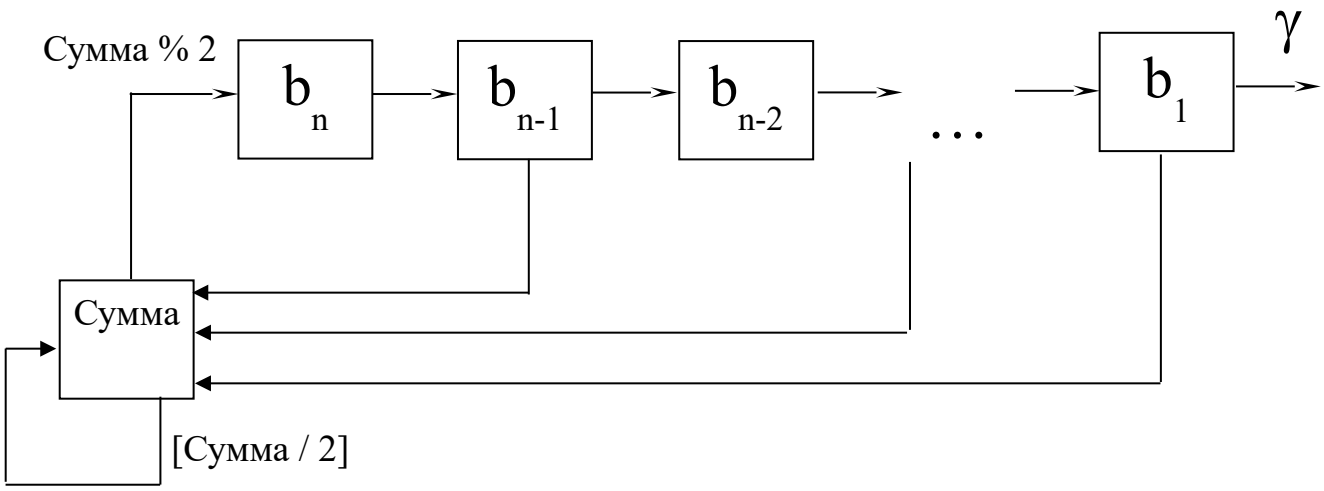


Рис.8. Сдвиговый регистр с обратной связью по переносу

В регистрах данного типа значение младшего бита формируется после суммированием всех бит обратной связи и содержимого регистра переноса. Остаток от деления на 2 получившейся суммы записывается в младший бит регистра, а результат деления нацело – в регистр переноса. Размер регистра переноса в битах должен быть равен $[\log_2 t\,]$, где t – количество ответвлений обратной связи.

Рассмотрим принцип работы РОСП на примере 4 битового регистра со структурой, изображенной на рис. 9

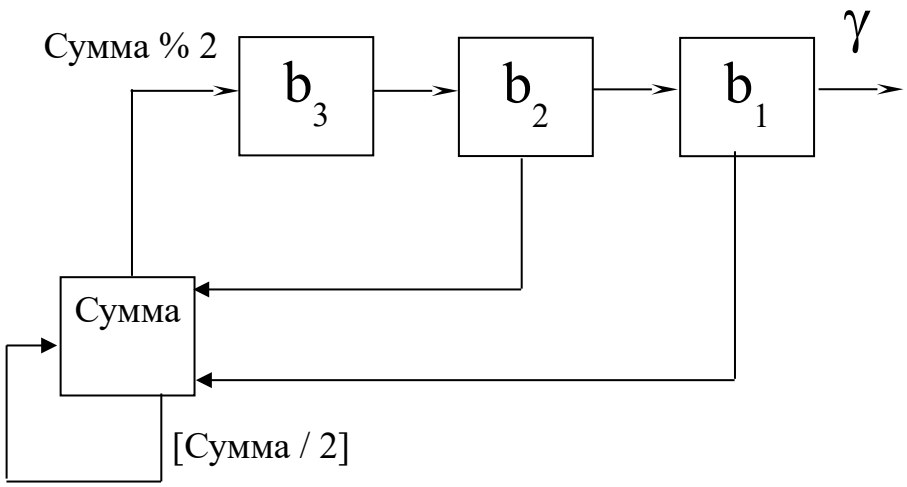


Рис.9. Регистр сдвига с обратной связью по переносу на 4 бита

Предположим, что его начальное состояния равно 101, а начальное состояние регистра переноса равно 1. Изменения внутреннего состояния РОСП и его выхода можно проследить по таблице 2.

Таблица 2. Результат работы генератора гаммы на основе РОСП

Номер такта	Значения битов РОСП			Регистр переноса	Бит гаммы
	3	2	1		
нач.сост.	1	0	1	1	
1	0	1	0	1	1
2	0	0	1	1	0
3	0	0	0	1	1
4	1	0	0	0	0
5	0	1	0	0	0
6	1	0	1	0	0
7	1	1	0	0	1
8	1	1	1	0	0
9	0	1	1	1	1
10	1	0	1	1	1

Максимальный период РОСП равен q-1, где q – целое число связи, его значение вычисляется по отводам обратной связи:

$$q = q_1*2 + q_2*2^2 + q_3*2^3 + \ldots + q_{n-1}*2^n - 1,$$

где q_i отсчитываются от левого края РОСП [5]. В связи с эти РОСП обозначают ($n_1, n_2, n_3, n_4, \ldots$), где n_i – номера тех разрядов, от которых строится обратная связь.

Необходимо отметить, что не все начальные состояния позволяют получить максимальный размер повторения гаммы. В связи с этим рекомендуется при выбранном начальном ключе (начальном состоянии РОСП) выполнить пробный запуск и, если поток гаммы не вырождается в бесконечный поток двоичных 0 (или 1), использовать данный ключ не практике.

Потоковые шифры

RC4 — это потоковый шифр, широко применяющийся в различных системах защиты информации в компьютерных сетях (например, в протоколе SSL и для шифрования паролей в Windows NT).Шифр разработан компанией RSA Security Inc. и для его использования требуется лицензия. Автором RC4 является Рональд Ривест (Ronald Rivest). Алгоритм RC4 строится, как и любой потоковый шифр, на основе параметризованного ключом генератора псевдослучайных битов с равномерным распределением. Основные преимущества шифра — высокая скорость работы и переменный размер ключа. Типичная реализация выполняет 19 машинных команд на каждый байт текста.

Ядро алгоритма состоит из функции генерации ключевого потока. Эта функция генерирует последовательность битов, которая затем объединяется с открытым текстом посредством суммирования по модулю два. Дешифрация состоит из регенерации этого ключевого потока и суммирования его с шифрограммой по модулю два, восстанавливая исходный текст. Другая главная часть алгоритма — функция инициализации, которая использует ключ переменной длины для создания начального состояния генератора ключевого потока.

RC4 — фактически класс алгоритмов, определяемых размером его блока. Этот параметр n является размером слова для алгоритма. Обычно, $n = 8$, но в целях анализа можно уменьшить его. Однако для повышения безопасности необходимо увеличить эту величину. Внутреннее состояние RC4 состоит из массива размером 2^n слов и двух счетчиков, каждый размером в одно слово. Массив известен как *S*-box, и далее будет обозначаться как *S*. Он всегда содержит перестановку 2^n возможных значений слова. Два счетчика обозначены через i и j.

Алгоритм инициализации RC4 приведен ниже. Этот алгоритм использует ключ *Key*, имеющий длину l байт. Инициализация начинается с заполнения массива *S*, далее этот массив перемешивается путем перестановок определяемых ключом.

Начальное заполнение массива *S*:

```
for i = 0 to 2^n – 1
        S [i] = i
```

Следующий этап – перестановка элементов *S*, параметризуемая ключом:

```
j = 0 for i = 0 to 2^n – 1:
     j=(j+S[i]+Key[i])mod2^n
     Перестановка (S [i] , S [j])
```

Генератор ключевого потока RC4 переставляет значения, хранящиеся в *S*, и каждый раз выбирает различное значение из *S* в качестве результата. В одном цикле RC4 определяется одно *n*-битное слово *K* из ключевого потока, которое в последующем суммируется с исходным текстом для получения зашифрованного текста.

Инициализация: `i = 0 j = 0`

Цикл генерации: `i = (i + 1) mod 2^n`

`j = (j + S [i] ) mod 2^n`

Перестановка `(S [i], S [j])`

Результат: `K = S [(S [i] + S [j] ) mod 2^n]`

RC4 получил широкое распространение в криптосистемах и протоколах, в частности:

- WEP (англ. Wired Equivalent Privacy) - алгоритм для обеспечения безопасности сетей Wi-Fi;
- WPA (англ. Wi-Fi Protected Access) - обновленный алгоритм сертификации устройств сетей Wi-Fi;
- BitTorrent protocol encryption - протоколы пиринговых файлообменных сетей;
- SSL (англ. Secure Sockets Layer) - криптографический протокол передачи данных в сети;
- Kerberos - сервер аутентификации Kerberos;
- PDF (англ. Portable Document Format) - межплатформенный формат электронных документов, разработанный фирмой Adobe Systems;
- Skype - программное обеспечение IP-телефонии;
- и др.

Алгоритм RC4 устойчив к дифференциальному и линейному криптоанализу: в нем нет коротких циклов, он нелинеен. При n=8 RC4 может находиться в примерно $2^{1700}(256! * 256^2)$ различных состояниях [5].

WAKE - сокращение от Word Auto Key Encryption (Автоматическое шифрование слов ключом). Алгоритм выдает поток 32-битовых слов, которые с помощью XOR могут быть использованы для получения шифротекста из открытого текста или открытого текста из шифротекста.

WAKE работает в режиме CFB, для генерации следующего слова ключа используется предыдущее слово шифротекста. Алгоритм также использует *S*-блок из 256 32-битовых значений. Содержимое *S*-блока наполняется по следующему принципу: старший байт всех элементов представляет собой перестановку всех возможных байтов, а в 3 младших байта заносятся случайные значения.

Генерация старших байтов элементов *S*-блока может быть выполнено аналогично алгоритму RC4. В младшие байты блоков S_i заносятся случайные значения.

Затем проинициализируем четыре регистра с использованием того же или иного ключа: a_0, b_0, c_0 и d_0. Поток ключей совпадает со значением регистра d_i: $K_i = d_i$.

Слово шифротекста C_i представляет собой XOR слова открытого текста P_i с K_i. На каждом шаге обновляются четыре регистра:

$$a_{i+i} = M(a_i, d_i)$$

$$b_{i+i} = M(a_i, d_i)$$

$$c_{i+i} = M(b_i, d_i)$$

$$d_{i+i} = M(c_i, d_i)$$

Функция M выполняет следующие преобразования:

$$M(x,y) = (x + y) >> 8 \oplus S_{(x + y) \bmod 255}$$

Самым ценным качеством WAKE является его скорость. Однако он чувствителен к вскрытию с выбранным открытым текстом или выбранным шифротекстом.

Trivium был представлен в 2008 г. как часть европейского проекта eSTREAM по профилю 2 (аппаратно-ориентированные шифры) и в настоящий момент имеет статус международного стандарта «ISO/IEC 29192-3:2012. Информационные технологии - Методы безопасности - Легкая криптография - Часть 3: Поточные шифры» (англ. «ISO/IEC 29192-3:2012. Information technology - Security techniques - Lightweight cryptography - Part 3: Stream ciphers»). Авторами генератора (шифра) являются Кристоф Де Канньер и Барт Пренел.

По аналогии с RC4 процедура выработки гаммы состоит из двух этапов: инициализации S-блока и непосредственно генерации гаммы.

Этап 1. Инициализация S-блока.

Длина S-блока составляет 288 битов. При этом он условно делится на 3 части длинами 93, 84 и 111 битов. В первую часть S_1 заносится 80-битовый ключ K с добавлением 13 нулевых битов, во вторую часть S_2 заносится 80-битовый вектор инициализации IV с добавлением 4 нулевых битов и в последнюю часть S_3 заносятся нулевые биты за исключением трех последних единичных битов. После первоначальной инициализации в цикле осуществляется связывание битов из разных частей со сдвигами битов вправо в каждой из них. Весь алгоритм инициализации выглядит следующим образом.

1. $S_1 = (s_1, s_2, ..., s_{93}) := (K_1, K_2, ..., K_{80}, 0, ..., 0)$

2. $S_2 = (s_{94}, s_{95}, ..., s_{177}) := (IV_1, IV_2, ..., IV_{80}, 0, ..., 0)$

3. $S_3 = (s_{178}, s_{179}, ..., s_{288}) := (0, ..., 0, 1, 1, 1)$

4. Цикл. Для i := 1 до 288

4.1. $t_1 := s_{66} \oplus (s_{91} \wedge s_{92}) \oplus s_{93} \oplus s_{171}$

4.2. $t_2 := s_{162} \oplus (s_{175} \wedge s_{176}) \oplus s_{177} \oplus s_{264}$

4.3. $t_3 := s_{243} \oplus (s_{286} \wedge s_{287}) \oplus s_{288} \oplus s_{69}$

4.4. Сдвиг первой части на один бит вправо $S_1 := (\_, s_1, s_2, ..., s_{92})$

4.5. Занесение t_3 в первую позицию первой части $S_1 := (t_3, s_1, s_2, ..., s_{92})$

4.6. Сдвиг второй части на один бит вправо $S_2 := (\_, s_{94}, s_{95}, ..., s_{176})$

4.7. Занесение t_1 в первую позицию второй части $S_2 := (t_1, s_{94}, s_{95}, ..., s_{176})$

4.8. Сдвиг третьей части на один бит вправо $S_3 := (\_, s_{178}, s_{179}, ..., s_{287})$

4.9. Занесение t_2 в первую позицию третьей части $S_3 := (t_2, s_{178}, s_{179}, ..., s_{287})$

Этап 2. Генерация гаммы.

Для генерации гаммы длиной L_g в каждой итерации цикла на основе 15 битов S-блока вычисляется один бит гаммы и выполняются операции, идентичные рассмотренным в инициализации S-блока. Алгоритм генерации следующий.

1. $L := 0$

2. Цикл. Пока $L < L_g$

2.1. Бит гаммы $:= s_{66} \oplus s_{93} \oplus s_{162} \oplus s_{177} \oplus s_{243} \oplus s_{288}$

2.2. $t_1 := s_{66} \oplus (s_{91} \wedge s_{92}) \oplus s_{93} \oplus s_{171}$

2.3. $t_2 := s_{162} \oplus (s_{175} \wedge s_{176}) \oplus s_{177} \oplus s_{264}$

2.4. $t_3 := s_{243} \oplus (s_{286} \wedge s_{287}) \oplus s_{288} \oplus s_{69}$

2.5. Сдвиг первой части на один бит вправо $S_1 := (\_, s_1, s_2, ..., s_{92})$

2.6. Занесение t_3 в первую позицию первой части $S_1 := (t_3, s_1, s_2, ..., s_{92})$

2.7. Сдвиг второй части на один бит вправо $S_2 := (\_, s_{94}, s_{95}, ..., s_{176})$

2.8. Занесение t_1 в первую позицию второй части $S_2 := (t_1, s_{94}, s_{95}, ..., s_{176})$

2.9. Сдвиг третьей части на один бит вправо $S_3 := (\_, s_{178}, s_{179}, ..., s_{287})$

2.10. Занесение t_2 в первую позицию третьей части $S_3 := (t_2, s_{178}, s_{179}, ..., s_{287})$

2.11. L := L + 1

Существуют упрощенные модификации генератора гаммы Univium, Bivium, Trivium-toy и Bivium-toy.

Порядок выполнения работы

1. Ознакомьтесь с теоретическими основами блочного и поточного симметричного шифрования в настоящих указаниях, а также в [2] и конспектах лекций.
2. *Получите* ***вариант*** задания у преподавателя.
3. Напишите *программу* согласно варианту задания.
4. Отладьте разработанную программу и покажите результаты работы программы преподавателю.
5. Составьте отчет по лабораторной работе.

Содержание отчета

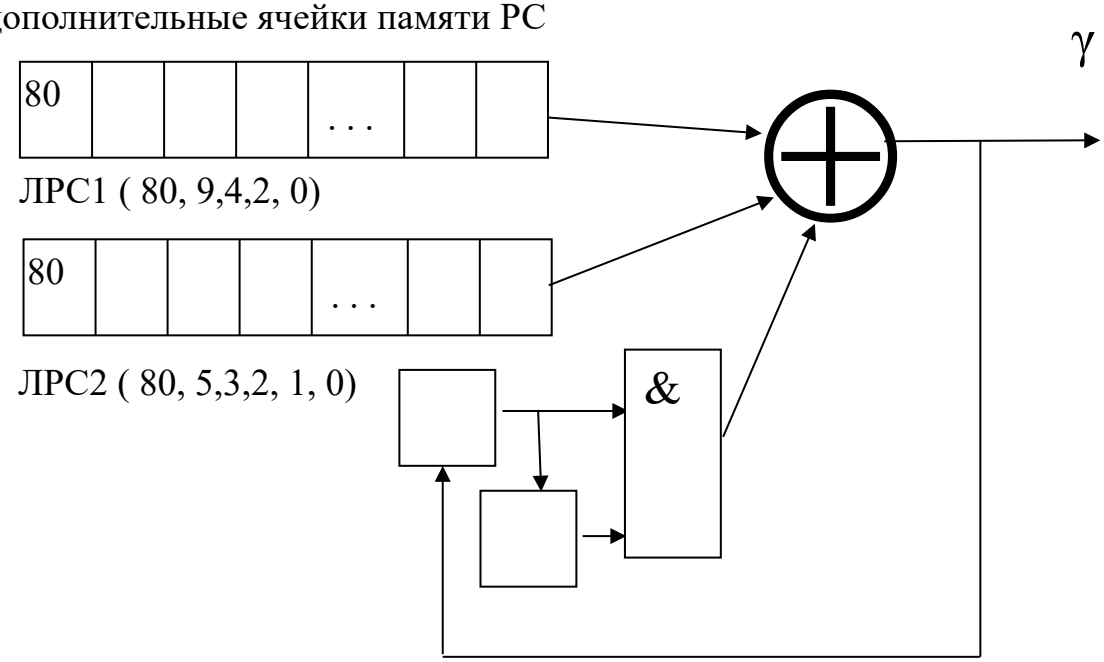
Отчет по лабораторной работе должен содержать следующие сведения:

- название и цель работы;
- вариант задания;
- листинг разработанной программы с комментариями;
- результаты работы программы.

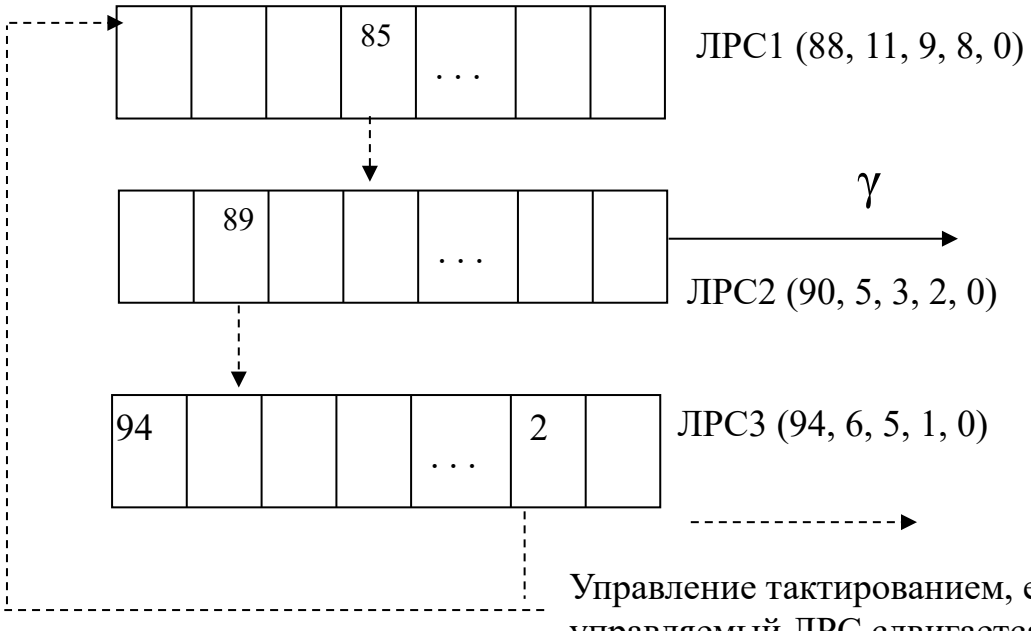
! Напомним, что короткая запись РСЛОС(8, 4, 3, 2, 0) или запись $x^8 + x^4 + x^3 + x^2 + 1$ означает, что длина регистра 8 битов, а входной бит рассчитывается по формуле $b_7 = b_4 \oplus b_3 \oplus b_2 \oplus b_0$ (для битов нумерация начинается справа и с нулевой позиции). Выходной (выдвигаемый справа, младший) бит будет являться частью генерируемой гаммы.

Варианты заданий (*задание 1 и задание 2*)

1. Реализовать в программе поточное кодирование текста, вводимого с клавиатуры, с помощью заданной нелинейной схемы, испльзующей дополнительные ячейки памяти РС

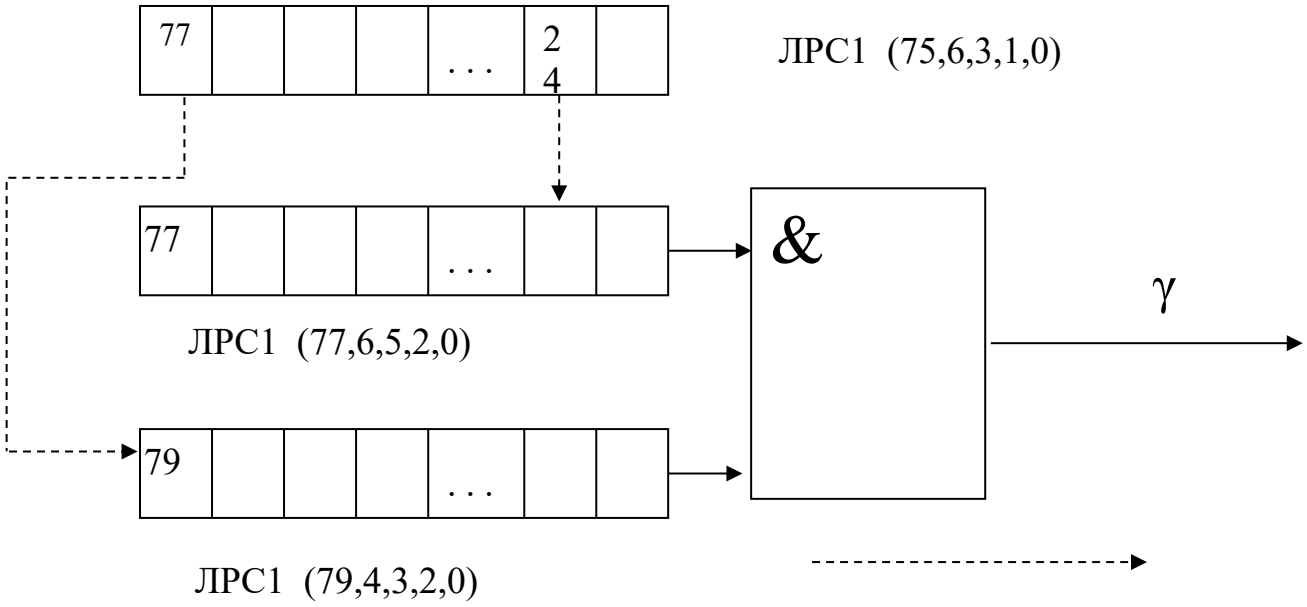


2. Реализовать в программе поточное кодирование текста, вводимого с клавиатуры, с помощью заданной нелинейной схемы РС с управляемым тактированием

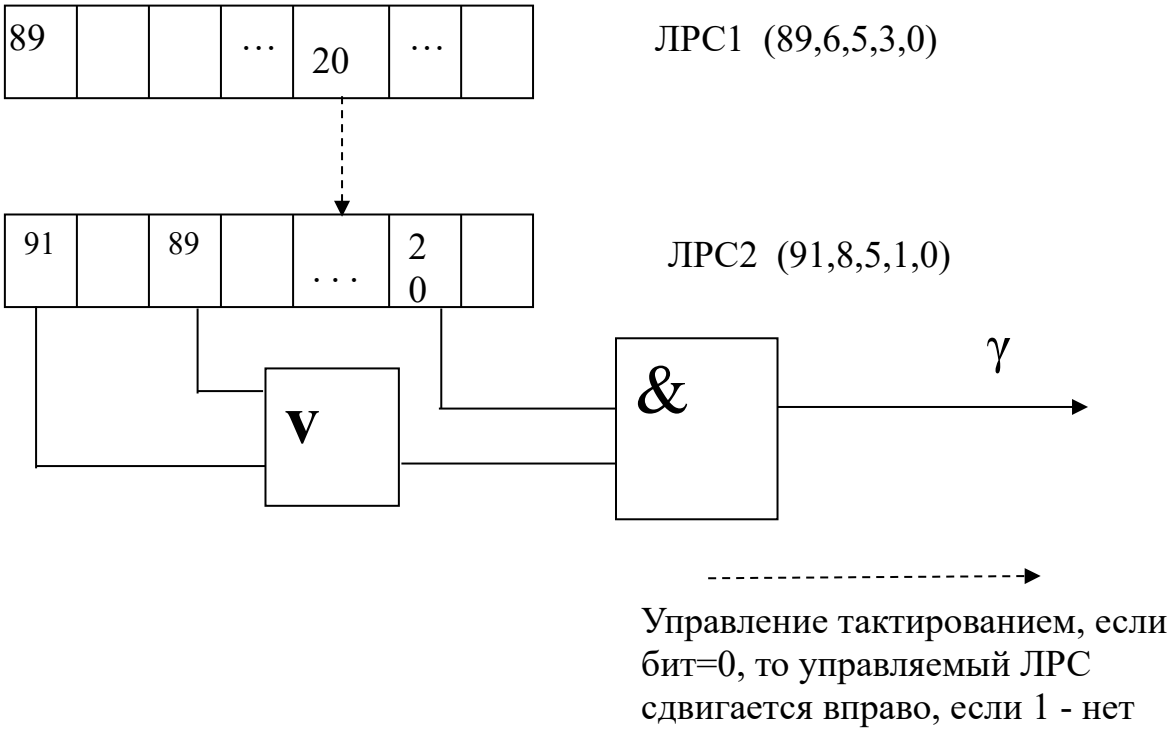


Управление тактированием, если бит=0, то управляемый ЛРС сдвигается вправо, если 1 - нет

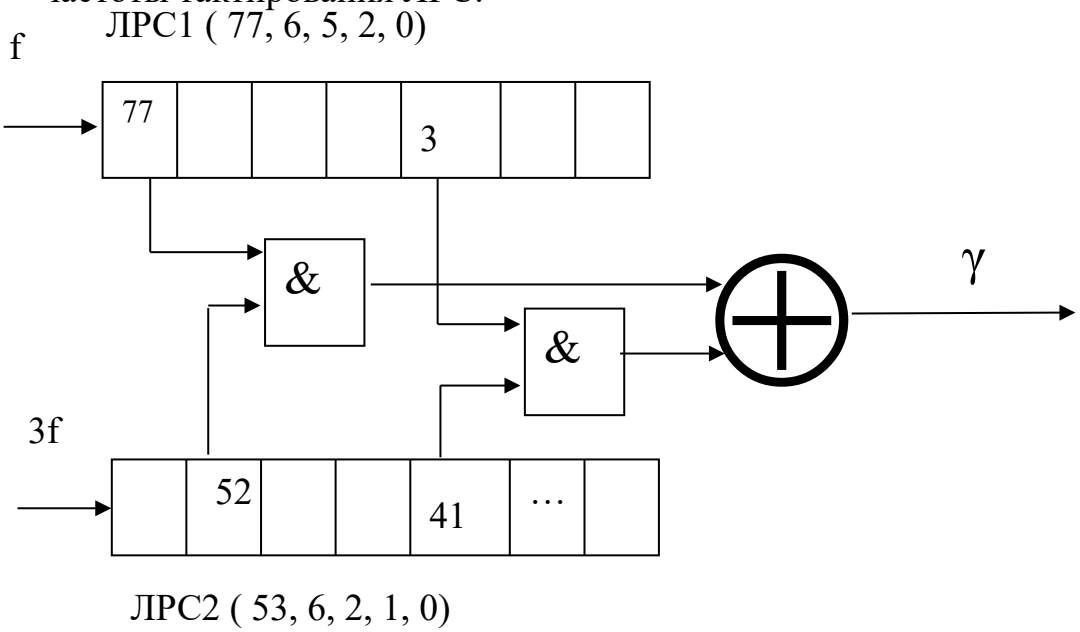
3. Реализовать в программе поточное кодирование текста, вводимого с клавиатуры, с помощью заданной нелинейной схемы РС



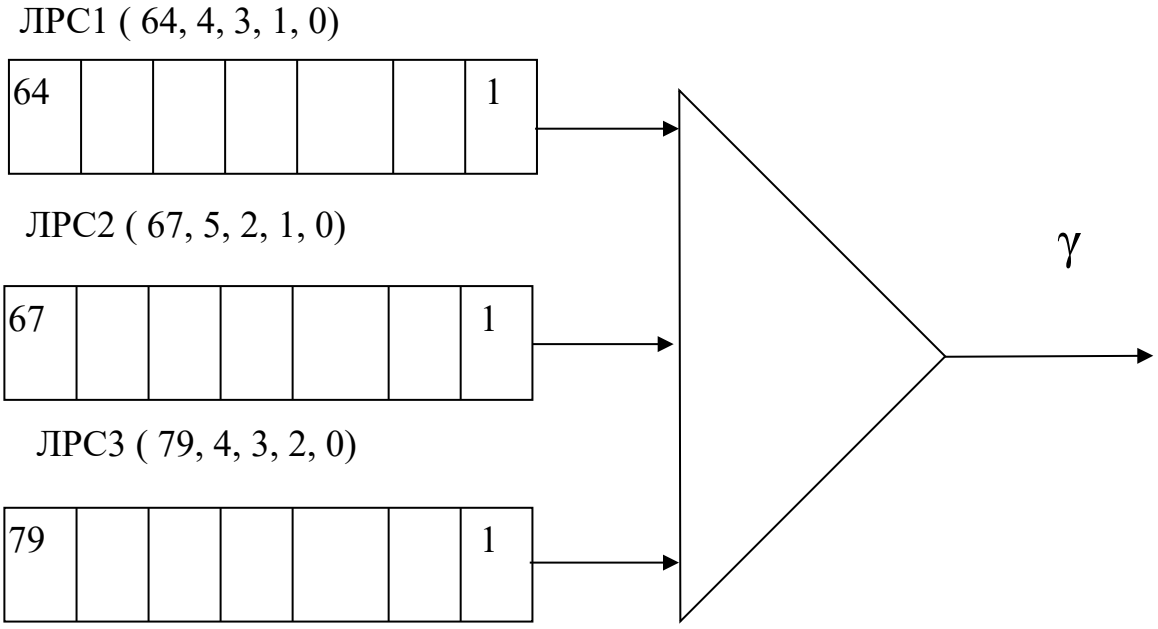
4. Реализовать в программе поточное кодирование текста, вводимого с клавиатуры, с помощью заданной нелинейной схемы РС



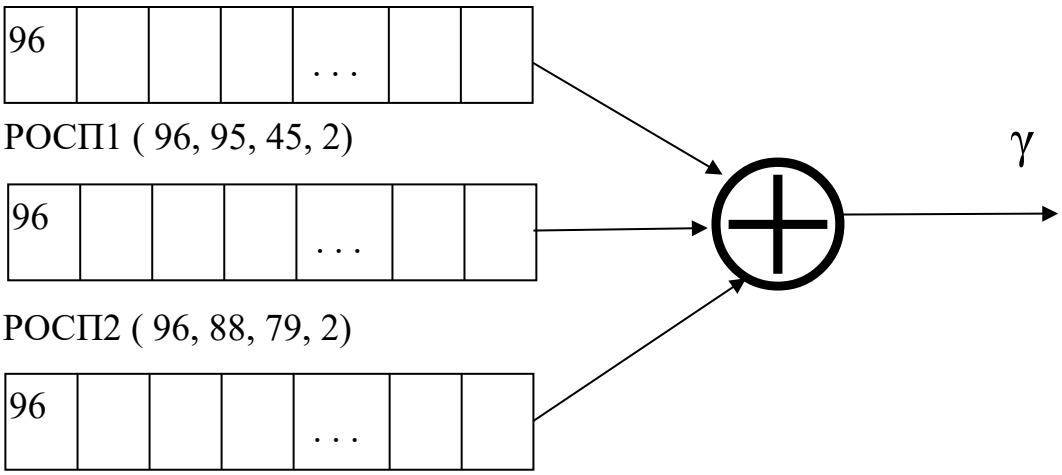
5. Реализовать в программе поточное кодирование текста, вводимого с клавиатуры, с помощью заданной нелинейной схемы, использующей разные частоты тактирования ЛРС.



6. Реализовать в программе поточное кодирование текста, вводимого с клавиатуры, с помощью заданной нелинейной схемы, использующей пороговую функцию – если выход двух и более ЛРС 1, то выход гаммы равен 1, иначе – 0.

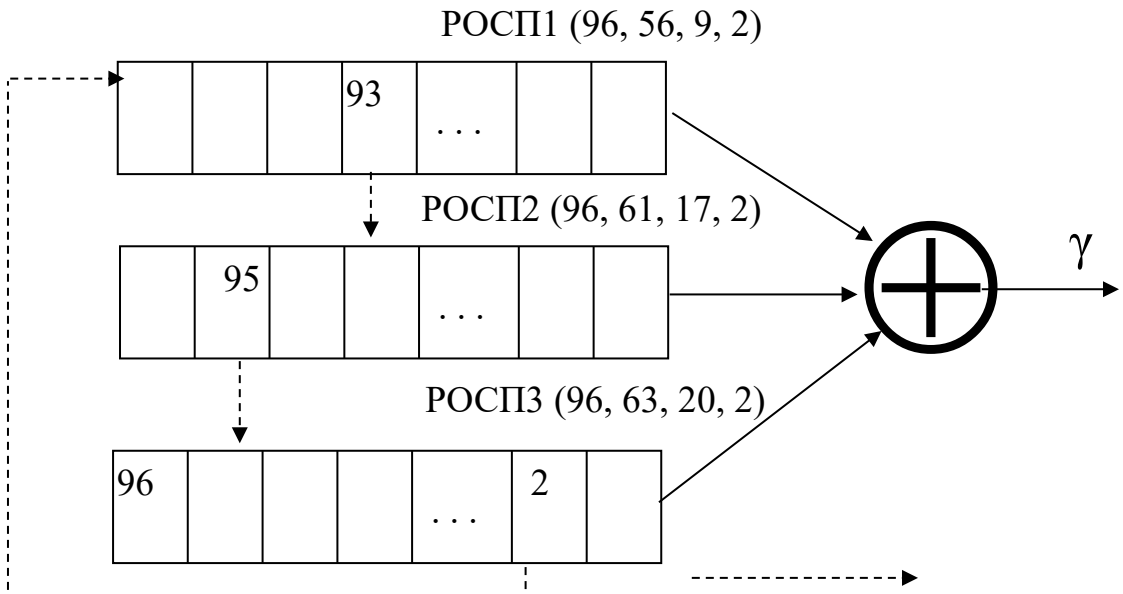


7. Реализовать в программе поточное кодирование текста, вводимого с клавиатуры, с помощью заданной схемы, объединяющей три регистра с обратной связью по переносу



РОСП3 (96, 69, 17, 2)

8. Реализовать в программе поточное кодирование текста, вводимого с клавиатуры, с помощью заданной схемы с управляемым тактированием на основе 3 РОСП



Управление тактированием, если бит=0, то управляемый ЛРС сдвигается вправо, если 1 - нет

Номер варианта определяется по след формуле ***(номер в списке)%8+1***

задание 2.

Программирование поточных шифров.

1, 5, 9, 13, 17, 21, 25. Реализовать в программе поточное кодирование текста, вводимого с клавиатуры, с помощью алгоритма RC4 с размером блока n=16 бит.

2, 6, 10, 14, 18, 22, 26. Реализовать в программе поточное кодирование текста, вводимого с клавиатуры, с помощью алгоритма WAKE.

3, 7, 11, 15, 19, 23, 27. Реализовать в программе поточное кодирование текста, вводимого с клавиатуры, с помощью алгоритма A5.

4, 8, 12, 16, 20, 24, 28. Реализовать в программе поточное кодирование текста, вводимого с клавиатуры, с помощью алгоритма Trivium.

Контрольные вопросы

1. Какие методы формирования потока ключей для поточных шифров Вам известны?
2. Что такое регистр сдвига с линейной обратной связью?
3. Каков критерий оптимальности структуры регистра сдвига с линейной обратной связью?
4. Для чего регистры сдвига с линейной обратной связью объединяют в нелинейные схемы подключения?
5. Что такое проблемы линейной сложности и корреляционной связи схем, использующих сдвиговые регистры с линейной обратной связью.
6. Объясните принцип работы сдвигового регистра с обратной связью по переносу.
7. Каков критерий оптимальности структуры регистра сдвига с обратной связью по переносу?

Рекомендуемые источники

1. Алферов А.П., Зубов А.Ю., Кузьмин А.С. Основы криптографии. - М: Гелиос АРВ, 2005 г. - 480 с.
2. Лясин Д.Н., Саньков С.Г. Методы и средства защиты компьютерной информации (учебное пособие). – Волгоград, Издательство ВолгГТУ РПК "Политехник”, 2005г.
3. Смарт Н. Криптография. – М: Техносфера, 2006 г. - 528 с.
4. Чмора А.Л. Современная прикладная криптография. 2-е изд. -М.: Гелиос АРВ, 2002.- 256с.:ил.
5. Шнайер Б. Прикладная криптография.- М. : Триумф, 2002. – 816с.