Агафонова Ирина Витальевна,
Дмитриева Оксана Михайловна

ДВОИЧНЫЕ ЛИНЕЙНЫЕ РЕКУРРЕНТНЫЕ ПОСЛЕДОВАТЕЛЬНОСТИ И АЛГОРИТМ БЕРЛЕКЭМПА-МЭССИ

Сообщения, передаваемые двоичным кодом по каналам связи, имеют вид последовательностей из нулей и единиц. Эта статья посвящена двоичным последовательностям, которые называют линейными рекуррентными последовательностями (ЛРП).

Каждый член линейной рекуррентной последовательности, начиная с некоторого номера, выражается в виде линейной комбинации определённого числа L своих непосредственных предшественников. Число L – это порядок последовательности. Например, известная последовательность Фибоначчи 1, 1, 2, 3, 5, 8, 13, … является линейной рекуррентной порядка 2: каждый ее член, начиная с третьего, есть сумма двух предыдущих. Нетрудно убедиться, что геометрическая и арифметическая прогрессии – тоже линейные рекуррентные последовательности порядков 1 и 2 соответственно.

При шифровании данных широко используются регистры сдвига с линейной обратной связью, порождающие линейные рекуррентные последовательности из нулей и единиц. О таких регистрах мы подробно рассказывали в [1]. Напомним, что каждый член выходной последовательности $s_0, s_1, s_2, \ldots$ регистра длины L, начиная с s_L, вычисляется как

$$s_i = c_L s_{i-L} \oplus c_{L-1} s_{i-L+1} \oplus \ldots \oplus c_2 s_{i-2} \oplus c_1 s_{i-1}, \quad i = L, L+1, \ldots, \qquad (1)$$

при заданном начальном заполнении $s_0, s_1, \ldots, s_{L-1}$ и определённом наборе коэффициентов $c_1, c_2, \ldots, c_L$.

Все значения, входящие в формулу (1), равны 0 или 1. В формуле используются операции логического умножения ($\cdot$ или AND, знак умножения часто опускается) и логического исключающего сложения ($\oplus$ или XOR).

Из формулы (1) видно, что выходная последовательность описанного регистра сдвига – линейная рекуррентная.

Конечно, двоичную линейную рекуррентную последовательность $s_0, s_1, s_2, \ldots$ можно создать, не обращаясь к понятию линейного регистра сдвига. А именно:

1. Выбирается целое число $L > 0$ – *порядок* ЛРП. Например, $L = 4$.
2. Берутся L произвольных значений 0 или 1 – *начальный вектор* ЛРП $(s_0, s_1, \ldots, s_{L-1})$. Например, $(s_0, s_1, s_2, s_3) = 1011$.
3. Задаётся набор коэффициентов $c = (c_1, c_2, \ldots, c_L)$, также равных 0 или 1. Например, $c = (c_1, c_2, c_3, c_4) = 0101$.
4. Все следующие члены последовательности вычисляются по формуле (1) один за другим:

$$s_L = c_L s_0 \oplus c_{L-1} s_1 \oplus \ldots \oplus c_2 s_{L-2} \oplus c_1 s_{L-1},$$
$$s_{L+1} = c_L s_1 \oplus c_{L-1} s_2 \oplus \ldots \oplus c_2 s_{L-1} \oplus c_1 s_L,$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$

Например,

$$s_4 = 1 \cdot s_0 \oplus 0 \cdot s_1 \oplus 1 \cdot s_2 \oplus 0 \cdot s_3 = 1 \cdot 1 \oplus 0 \cdot 0 \oplus 1 \cdot 1 \oplus 0 \cdot 1 = 0,$$
$$s_5 = 1 \cdot s_1 \oplus 0 \cdot s_2 \oplus 1 \cdot s_3 \oplus 0 \cdot s_4 = 1 \cdot 0 \oplus 0 \cdot 1 \oplus 1 \cdot 1 \oplus 0 \cdot 0 = 1.$$

Последовательность, определяемая соотношением (1), – бесконечная. Её начальную часть $s_0, s_1, s_2, \ldots, s_{n-1}$ при любом $n \geq L$ будем называть *отрезком* данной ЛРП длины n. Отрезок длины L совпадает с начальным вектором.

Добавим к набору коэффициентов ещё один коэффициент c_0, всегда равный 1. Тогда получаем запись формулы (1) в виде

$$c_L s_{i-L} \oplus c_{L-1} s_{i-L+1} \oplus \ldots \oplus c_2 s_{i-2} \oplus c_1 s_{i-1} \oplus c_0 s_i = 0, \quad i = L, L+1, \ldots \qquad (2)$$

Если дана какая-то конечная последовательность из n битов

$$s_0, s_1, s_2, \ldots, s_{n-1}, \qquad (3)$$

то по формуле (2) удобно проверить, будет ли она отрезком линейной рекуррентной последовательности порядка $L \leq n$ с набором коэффициентов $c_1, c_2, \ldots, c_L$.

Левая часть формулы (2), которую мы обозначим

$$d(i) = c_L s_{i-L} \oplus c_{L-1} s_{i-L+1} \oplus \ldots \oplus c_2 s_{i-2} \oplus c_1 s_{i-1} \oplus c_0 s_i, \qquad (4)$$

носит название «бит несоответствия». Если при каком-то i из диапазона $L \leq i \leq n + L - 1$ этот бит равен 1, то последовательность (3) не является линейной рекуррентной порядка L.

Проверка выполнимости формулы (2) будет чисто механическим действием, если представить себе (подобный приём применялся в [1] для линейных регистров сдвига), что мы короткую линейку

c_L	c_{L-1}	…	c_2	c_1	c_0=1

прикладываем к длинной линейке

s_0	s_1	…	s_{L-2}	s_{L-1}	s_L	…	s_{n-2}	s_{n-1}

При этом сначала мы c_L совмещаем с s_0, соответственно c_{L-1} совместится с s_1 и так далее, c_0 совместится с s_L. Теперь перемножаем соответствующие элементы и складываем результаты по XOR. Должен получиться 0.

Затем сдвигаем короткую линейку на одно деление вправо (c_L совмещаем с s_1, при этом c_0 совместится с s_{L+1}) и повторяем проверку.

Затем ещё раз сдвигаемся вправо и так далее. Последним будет совмещение c_L и s_{n-L-1}, при этом c_0 совместится с s_{n-1}. При каждой проверке должны получаться нули.

Усложним задачу. Мы по-прежнему имеем последовательность из n битов вида (3), но теперь мы не располагаем информацией о значениях порядка и вектора коэффициентов.

Требуется выяснить, каким регистром сдвига порождена данная последовательность (3), или, что то же самое, отрезком какой ЛРП она является. При этом нужно найти порядок L и вектор коэффициентов c.

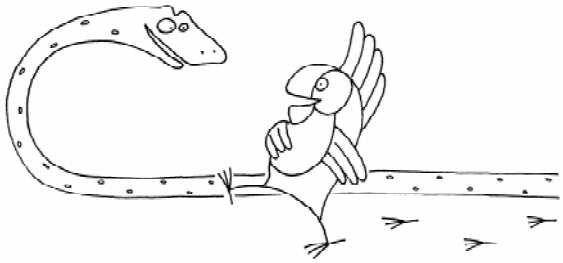
Сразу надо сказать, что одна и та же конечная битовая последовательность может порождаться разными регистрами и, соответственно, по-разному определяться как ЛРП. Например, последовательность 01010101 является отрезком:

– линейной рекуррентной последовательности порядка 4 с вектором коэффициентов $c = (c_1, c_2, c_3, c_4) = (0, 0, 0, 1)$ и начальным вектором $(s_0, s_1, s_2, s_3) = (0, 1, 0, 1)$;

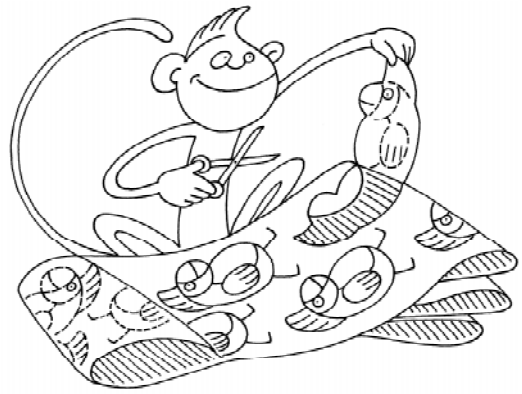
– линейной рекуррентной последовательности порядка 2 с вектором коэффициентов $c = (c_1, c_2) = (0, 1)$ и начальным вектором $(s_0, s_1) = (0, 1)$.

Наименьший порядок ЛРП, отрезком которой является конечная последовательность (3), называют *линейной сложностью* последовательности (3). Линейная сложность последовательности 01010101 равна 2.

Существует алгоритм, который по заданной последовательности (3) определяет её линейную сложность L и коэффициенты соответствующей ЛРП порядка L. Он называется *алгоритмом Берлекэмпа-Мэсси*.



...представим себе, будто мы короткую линейку прикладываем к длинной линейке...



Требуется выяснить, каким регистром сдвига порождена данная последовательность...

Для начала работы алгоритма предлагается найти первый ненулевой бит последовательности (3). Пусть это бит s_j, то есть

$$s_j = 1,$$

$$s_i = 0 \text{ при } i < j.$$

Другими словами, любая последовательность (3) начинается с отрезка

$$00\ldots01, \qquad (5)$$

длины $j + 1$, первые j битов которой нулевые. В частности, при $j = 0$ отрезок (5) состоит только из $s_0 = 1$.

Если принять $L = j + 1$, то отрезок (5) будет начальным вектором последовательности (3). Начнем с коэффициентов $c_0 = 1$, $c_1 = c_2 = c_3 = \ldots = 0$, $c_L = 1$, при этом для отрезка (5) формула (2) выполняется.

Алгоритм будет шаг за шагом считывать s_i, беря сначала $i = j + 1$, затем $i = j + 2$ и так далее. На каждом шаге алгоритм при необходимости будет изменять коэффициенты и, возможно, порядок L, так чтобы на отрезке

$$s_0, s_1, s_2, \ldots, s_i$$

длины $i + 1$ по-прежнему выполнялось условие ЛРП (формула (2)).

Если на шаге i при имеющихся c и L выполняется условие (2), то ничего менять не надо.

Значение L по ходу алгоритма время от времени может увеличиваться, но во всяком случае не превысит n – длины последовательности (3). Длину $n + 1$ (с учётом c_0) мы зарезервируем для массива c, чтобы не наращивать её постоянно, а «лишние» биты $c_{L+1}, c_{L+2}, \ldots, c_n$ заполним нулями.

Приведём детальное описание алгоритма Берлекэмпа-Мэсси. Будем сопровождать его числовым примером (рис. 1).

Значения L и c, полученные по окончании $(n – 1)$-го шага, дают решение задачи. В данном примере после 7-го шага будем иметь линейную сложность $L = 4$ и коэффициенты $c_1 = 0, c_2 = 0, c_3 = 1, c_4 = 1$. Проверить полученное решение можно, применив формулу (1) для $L = 4$ к отрезку 0101 при найденных коэффициентах c_i, $i = 1, 2, 3, 4$.

Для самостоятельного решения

1. Довести до конца решение приведённого примера.

2. Восстановить линейную сложность и коэффициенты ЛРП по её отрезку: 1110100111. Сделать проверку.

Теоремы, на которых основывается алгоритм Берлекэмпа-Мэсси, можно найти в книгах [2, 3, 4].

Замечание

Описанный алгоритм будет работать с любой битовой последовательностью (3). Но следует иметь в виду, что иногда полученная линейная сложность составит больше половины длины этой последовательности. Например, для 100001 будет получено значение $L = 5$ при $c_1 = c_2 = c_3 = c_4 = 0, c_5 = 1$. $c = 10001$. Это значит, что первые 5 значений 10000 должны быть приняты в качестве начального вектора ЛРП, и только последнее значение мы тогда можем получить по рекуррентной формуле (1).

Ещё меньше полезной информации мы получим, если применим алгоритм Берлекэмпа-Мэсси к последовательности с единственной последней единицей, например, к 0001. Мы получим $L = 4$, так что, если нам угодно, мы можем считать 0001 отрезком ЛРП, но только начальным.

В литературе обычно подчёркивается, что алгоритм полностью определяет характеристики последовательности (3) или, другими словами, вычисляет порождающий её регистр сдвига длины L, если длина самой последовательности (3) не меньше $2L$. На-

Начало.

	Пример
Имеем: последовательность (3) длины n.	01011110, $n = 8$
Вычисляем: j – первый индекс такой, что $s_j = 1$.	$s_0 = 0$, $s_1 = 1$, $j = 1$
Полагаем: $L = j + 1$ – порядок ЛРП, начальное значение	$L = 2$
c – массив битов c_i , $I = 0, 1, \ldots, n$, в котором полагаем $c_0 = c_L = 1$, остальные $c_i = 0$.	При ручном счёте массив удобно записывать справа налево: $\overleftarrow{c} = (0,0,0,0,0, 0, c_2, c_1, c_0) = (0,0,0,0,0,0,1,0,1)$, где $\overleftarrow{c}$ означает, что компоненты вектора с записаны по убыванию индексов. Краткая запись: $\overleftarrow{c} = 101$
$m = j$ – вспомогательная переменная, впоследствии это будет номер шага, на котором был последний скачок величины L.	$m = 1$
b – вспомогательный массив битов b_i , $i = 0, 1, \ldots, n$, в котором полагаем $b_0 = 1$, остальные $b_i = 0$; впоследствии это будет состояние массива c перед шагом m.	Как и массив c, для ручного счета записываем b справа налево: $\overleftarrow{b} = (b_8, b_7,\ldots, b_1, b_0) = (0,0,0,0,0,0,0,0,1)$
t – вспомогательный массив битов $(t_0, t_1, \ldots, t_n)$ для запоминания промежуточных массивов	
$j + 1$ – номер первого реального шага, предыдущие j шагов считаем пустыми.	$j + 1 = 2$

На i-м шаге, $i = j + 1, j + 2, \ldots, n–1$ производим следующие действия:

		Пример, $i = 2$	Пример, $i = 3$	Пример, $i = 4$
1	Проверяем для i соотношение (4), вычисляя бит несоответствия $d(i) = c_L s_{i–L} \oplus \ldots \oplus c_1 s_{i–1} \oplus c_0 s_i$.	$d(2) = c_2 s_0 \oplus c_1 s_1 \oplus \oplus c_0 s_2 = 1 \cdot 0 \oplus 0 \cdot 1 \oplus \oplus 1 \cdot 0 = 0$.	$d(3) = c_2 s_1 \oplus \oplus c_1 s_2 \oplus c_0 s_3 = = 1 \cdot 1 \oplus 0 \cdot 0 \oplus \oplus 1 \cdot 1 = 0$.	$d(4) = c_2 s_2 \oplus c_1 s_3 \oplus c_0 s_4 = = 1 \cdot 0 \oplus 0 \cdot 1 \oplus 1 \cdot 1 = 1$.
2	Если $d(i) = 0$, переходим к п.6, порядок менять не надо.	Переходим к п.6.	Переходим к п.6.	
3	При $d(i) = 1$ запоминаем $t{:=}c$ и меняем c: $c := c \oplus b'$ (побитовое логическое сложение), где b'– массив, полученный сдвигом массива b на $i – m$ позиций в сторону возрастания индексов: $b'_{k+i–m} = b_k$ при $k = 0, 1, 2, \ldots,$ $b'_0 = b'_1 = \ldots = b'_{i+m–1} = 0$.			$d(4) = 1$, поэтому запоминаем $\overleftarrow{t} = (0,0,0,0,0,0,1,0,1)$. Вычисляем $\overleftarrow{b}'$ как сдвиг $\overleftarrow{b}$ на $i – m = 4 – 1 = 3$ позиции влево, справа дополняя нулями. $\overleftarrow{b}' = (0,0,0,0,0,1,0,0,0)$. Меняем $\overleftarrow{c} := \overleftarrow{c} \oplus \overleftarrow{b}' = = (0,0,0,0,0,1,1,0,1)$.
4	Проверяем, нужно ли увеличивать значение L: в новом массиве с могут появиться единицы в позициях с номерами, большими, чем L. Оказывается, что порядок L должен возрасти, если имеет место неравенство $2L \le i$. Если $2L > i$, порядок оставляем прежним и переходим к п.6.			$L = 2$, $i = 4$. Так как $2L \le i$, нужно будет менять L.
5	Изменяем порядок $L := i + 1 – L$ (это происходит при $2L \le i$) и запоминаем i-й шаг как шаг со скачком значения L $m := i$, $b := t$.			Полагаем $L := i + 1 – L = 3$, $m := 4$, $\overleftarrow{b} := (0,0,0,0,0,0,1,0,1)$.
6	Конец шага i.	Конец шага 2.	Конец шага 3.	Конец шага 4.

Рис. 1

помним, что заранее величина L не известна, так что просто берётся (3) достаточно большой длины n, и, если после расчёта оказывается, что $n \geq 2L$, задача решена.

Линейные рекуррентные последовательности используются в поточном шифровании для создания ключевой последовательности γ. О поточном шифровании мы уже говорили в статье [1]. Процесс шифрования выглядит так:

– отправитель располагает секретным ключом (конечный набор нулей и единиц) . Используя этот ключ, он генерирует ключевую последовательность γ (сколь угодно длинную). Её часто так и называют *гаммой*.

– пусть α – исходное (незашифрованное) сообщение, состоящее из нулей и единиц, β – сообщение, полученное после шифрования. Формула перехода от α к β:

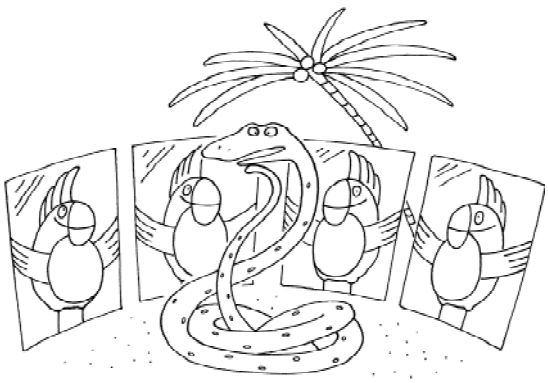
$$\beta = \alpha \oplus \gamma.$$

Эта запись означает, что каждый бит сообщения β получается сложением по XOR соответствующих битов исходного сообщения α и отрезка ключевой последовательности γ той же длины, что и α.

– получатель имеет тот же секретный ключ и генерирует ту же гамму. Он получает сообщение β и расшифровывает его по формуле

$$\alpha = \beta \oplus \gamma.$$

Способы генерирования последовательности γ могут быть различными. Совсем просто было бы использовать секретный ключ как начальный вектор $s_0, s_1, \ldots, s_{L-1}$, при некотором наборе коэффициентов $c_1, c_2, \ldots, c_L$ получить линейную рекуррентную последовательность и взять её в качестве гаммы. К сожалению, ЛРП как ключевая последовательность ненадёжна.



Этот момент шифрования называется запутыванием, и мы опишем простейший способ такого запутывания — комбинирующий генератор.

Действительно, если γ – ЛРП и если мы перехватили достаточно большой фрагмент исходного текста α и этого же зашифрованного текста $\beta = \alpha \oplus \gamma$, то легко получить отрезок последовательности $\gamma = \alpha \oplus \beta$. По этому отрезку, который мы можем записать как (3), уже знакомый нам алгоритм Берлекэмпа-Мэсси определит линейную сложность ЛРП, её начальный вектор и вектор коэффициентов.

После этого мы можем сами порождать гамму, накладывать её на зашифрованный текст β и получать исходный текст α, то есть мы вскрыли шифр.

Теперь нас не должно удивлять, что если мы привлекаем ЛРП в качестве инструмента для шифрования, то должен существовать какой-то переход от ЛРП к ключевой последовательности γ. Этот момент шифрования называется запутыванием, и мы опишем простейший способ такого запутывания – *комбинирующий генератор*.

Этот способ создаёт гамму из нескольких ЛРП, используя *булевы функции* $f(x_1, x_2, \ldots, x_k)$, то есть функции k переменных, в которых каждый из аргументов x_j и значения f принадлежат двухэлементному множеству $\{0, 1\}$.

Любую булеву функцию можно задать, перечислив её значения при всех возможных значениях её аргументов. Например, известная функция – дизъюнкция двух переменных (OR) – задаётся так:

$$f(0, 0) = 0,$$
$$f(0, 1) = 1,$$
$$f(1, 0) = 1,$$
$$f(1, 1) = 1.$$

Булевы функции можно задавать также и формулами. Одной из таких формул является представление булевой функции в виде двоичного многочлена от нескольких переменных с использованием операций AND и

XOR, причём знак логического умножения опускается. Та же функция двух переменных OR может быть задана формулой $f(x_1, x_2) = x_1 \oplus x_2 \oplus x_1 x_2$.

Такое представление называется *алгебраической нормальной формой* (АНФ) данной функции или *полиномом Жегалкина*. Степень этого полинома, то есть число сомножителей в самом длинном логическом произведении называют *алгебраической степенью* булевой функции. Дизъюнкция в нашем примере – функция степени 2.

Отметим, что так как логическое умножение определяется правилами

$$0 \cdot 0 = 0,\ 0 \cdot 1 = 0,\ 1 \cdot 0 = 0,\ 1 \cdot 1 = 1,$$

то всегда $x_i \cdot x_i = x_i$, так что в АНФ не бывает степеней переменных.

Булевы функции работают в комбинирующей модели следующим образом.

Выбрана функция $f(x_1, x_2, \ldots, x_k)$. Взяты k линейных рекуррентных последовательностей или, как это обычно представляют, взято k линейных регистров сдвига, порождающих такие последовательности. Каждый из регистров $j = 1, 2, \ldots, k$ на каждом такте i, где $i = 0, 1, 2, \ldots$, формирует очередное значение соответствующей ЛРП. Именно это значение на i-м такте принимается в качестве аргумента x_j, после чего вычисляют $\gamma_i = f(x_1, x_2, \ldots, x_k)$.

Схема комбинирующего генератора приведена на рис. 2.

Пример

Пусть четыре линейных регистра сдвига характеризуются следующими векторами коэффициентов и начальными состояниями:

1) $(c_2, c_1) = (1, 1)$, $(s_0, s_1) = (1, 0)$,
2) $(c_3, c_2, c_1) = (1, 0, 1)$, $(s_0, s_1, s_2) = (0, 1, 1)$,
3) $(c_3, c_2, c_1) = (0, 1, 1)$, $(s_0, s_1, s_2) = (1, 1, 0)$,
4) $(c_3, c_2, c_1) = (1, 1, 1)$, $(s_0, s_1, s_2) = (0, 0, 1)$.

Эти регистры вырабатывают (вспомним формулу (1)) четыре линейные рекуррентные последовательности, одну порядка 2 и три порядка 3. Мы можем выписать отрезки этих последовательностей длины, например 15:

1) 101101101101101,
2) 011101001110100,
3) 110110110110110,
4) 001100110011001,

Как уже отмечалось в [1], все ЛРП являются периодическими. В нашем случае

– 1-я ЛРП периодическая с самого начала, период 101;

– 2-я ЛРП периодическая, начиная с s_1, период 1110100;

– 3-я ЛРП периодическая с самого начала, период 110;

– 4-я ЛРП периодическая с самого начала, период 0011.

В качестве комбинирующей возьмём функцию $f(x_1, x_2, x_3, x_4) = x_2 \oplus x_1 x_4 \oplus x_1 x_2 x_3 x_4$. Алгебраическая степень этой функции равна 4: присутствует произведение четырёх переменных.

Тогда

$\gamma_0 = f(1, 0, 1, 0) = 0 \oplus 1 \cdot 0 \oplus 1 \cdot 0 \cdot 1 \cdot 0 = 0$,
$\gamma_1 = f(0, 1, 1, 0) = 1 \oplus 0 \cdot 0 \oplus 0 \cdot 1 \cdot 1 \cdot 0 = 1$

и так далее.

Первые 15 битов ключевой последовательности: 010101101111101.

Не все булевы функции пригодны для использования в описанной модели. Например, функция степени 1 (то есть не содержащая произведений переменных) будет неподходящей. Действительно, последовательность, порождаемая такой функцией, являясь суммой нескольких периодических последовательностей, тоже будет периодической (с периодом не больше, чем наименьшее общее кратное периодов складываемых последовательностей). Значит, такая комбинация регистров будет вести себя как один «большой» регистр, не защищенный от алгоритма Берлекэмпа-Мэсси. Чем выше алгебраическая степень булевой функции, тем сложнее получающаяся последовательность и тем труднее её вскрыть.

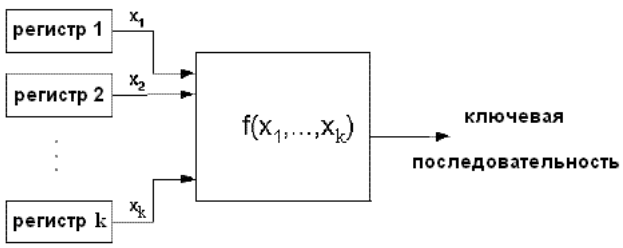


Рис. 2

Существуют и другие требования к булевым функциям, которые характеризуют устойчивость получаемой последовательности к тем или иным попыткам взлома. Приведём названия некоторых из них: уравновешенность, высокая корреляционная иммунность, высокая эластичность. Об этих и других криптографических характеристиках булевых функций можно прочесть в книге [4].

Интересно, что булева функция принципиально не может быть наилучшей по всем показателям. Это следует, в частности, из известных в криптографии неравенств вида $a + b \le \text{const}$, где a – один криптографический показатель, b – другой. Разумеется, оба показателя не могут одновременно быть высокими, так что при выборе функции для шифрования приходится искать компромисс.

Построение булевых функций с приемлемыми криптографическими характеристиками – актуальная и непростая задача. Поскольку необходимо противостоять все более мощным атакам, список требований, предъявляемых к булевым функциям, продолжает расширяться. Исследования в этой области основаны на математических знаниях, выходящих далеко за рамки школьной программы.

Литература

1. *Агафонова И.В., Дмитриева О.М.* Линейные регистры сдвига и поточные шифры // Компьютерные инструменты в школе, 2011. № 2.

2. *Х.К.А. ван Тилборг.* Основы криптологии. М.: Мир, 2006.

3. *Р. Блейхут.* Быстрые алгоритмы цифровой обработки сигналов. Пер. с англ. М.: Мир, 1989.

4. *Логачёв О.А., Сальников А.А., Яценко В.В.* Булевы функции в теории кодирования и криптологии. М.: МЦНМО, 2004.

Агафонова Ирина Витальевна, кандидат физико-математических наук, доцент кафедры исследования операций математико-механического факультета СПбГУ,

Дмитриева Оксана Михайловна, кандидат физико-математических наук, доцент кафедры высшей математики СПбГУТ.