Агафонова Ирина Витальевна,
Дмитриева Оксана Михайловна

ЛИНЕЙНЫЕ РЕГИСТРЫ СДВИГА И ПОТОЧНЫЕ ШИФРЫ

Мы расскажем здесь об одном устройстве, широко применяемом в криптографии. Слово «криптография» буквально может быть переведено с греческого как «секретное письмо»: отправитель желает, чтобы никто, кроме адресата, не смог прочесть послание.

Перевод открытого текста в зашифрованный с помощью секретного ключа называют шифрованием. Для того чтобы прочесть зашифрованное сообщение, достаточно знать этот ключ.

Мы будем считать, что передаваемое сообщение (цифры, латинские и русские буквы, математические знаки, рисунки и прочее) представлено в виде двоичного кода. Таким образом, отправитель посылает адресату последовательность из нулей и единиц. Например, в распространённой кодировке CP1251, используемой в Windows, строка *Агент007* будет иметь вид
10000000 10100011 10100101 10101101 11100010 00110000 00110000 00110111.
(Пробелы, отделяющие друг от друга коды символов, указаны здесь для наглядности. На самом деле в кодовой строке есть только нули и единицы.)

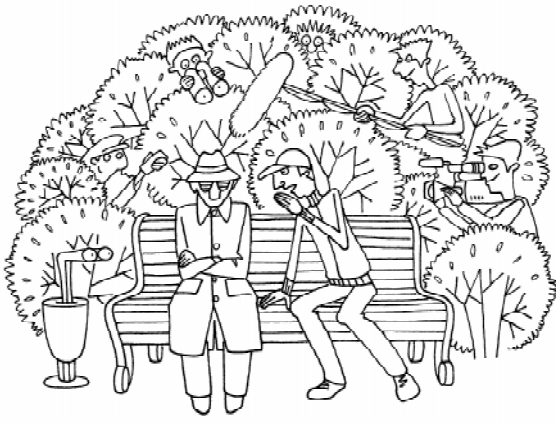
Секретный ключ также будет иметь двоичное представление.

БЛОЧНЫЕ И ПОТОЧНЫЕ ШИФРЫ

В случае, когда один и тот же секретный ключ используется и при шифровании, и при расшифровке, говорят, что мы имеем дело с симметричными криптосистемами. В этом случае обе стороны, обменивающиеся посланиями, должны знать этот ключ, и возникает проблема: как сообщить выбранный ключ партнёру?

Эта проблема снимается в асимметричных криптосистемах, появившихся в последней четверти XX века. В этих криптосистемах применяют два ключа: открытый ключ – для шифрования, а секретный – для расшифровки, так что отправителю знать секретный ключ не обязательно. Но шифрование с открытым ключом требует более сложных операций, чем традиционные симметричные шифры, и работает значительно медленнее.

Существует и комбинация симметричной и асимметричной схем шифрования, когда последняя используется только для передачи ключа.



...обе стороны... должны знать этот ключ, и возникает проблема: как сообщить выбранный ключ партнёру?

Большая часть симметричных шифров относится или к группе поточных шифров, когда каждый элемент входных данных (бит или символ) сразу же шифруется, или к группе блочных шифров, когда за один шаг обрабатывается блок элементов данных (например, 64 бита, как в стандарте ГОСТ). Напомним, что бит – это один двоичный разряд, базовая единица измерения информации.

Мы будем говорить только о поточных шифрах и о той роли, которую играют в поточном шифровании устройства, называемые регистрами сдвига с линейной обратной связью.

ЛОГИЧЕСКИЕ ОПЕРАЦИИ НАД ПОСЛЕДОВАТЕЛЬНОСТЯМИ

Введём над двоичными последовательностями одинаковой длины побитовые операции:

– логическое умножение (знак · или AND; этот знак часто будем опускать)

– логическое исключающее сложение (знак ⊕ или XOR).

Приведём таблицы умножения и исключающего сложения для битов:

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

$$0 \oplus 0 = 0,\ 0 \oplus 1 = 1,\ 1 \oplus 0 = 1,\ 1 \oplus 1 = 0.$$

...при сложении двух шифртекстов, соответствующих одному ключу, получается сумма исходных текстов, что... часто позволяет их прочесть.

При побитовых операциях каждый бит одной последовательности умножается на находящийся в той же позиции бит другой последовательности или складывается с этим битом. То есть для последовательностей

$$\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n),\ \beta = (\beta_1, \beta_2, \ldots, \beta_n),$$
$$\mu = (\mu_1, \mu_2, \ldots, \mu_n),$$

выражения $\mu = \alpha \cdot \beta$ или $\mu = \alpha \oplus \beta$ будут означать, что $\mu_i = \alpha_i \cdot \beta_i$ или $\mu_i = \alpha_i \oplus \beta_i$ для всех i от 1 до n.

Пример. Для двух последовательностей $\alpha = 10001$, $\beta = 00111$ длины 5 имеем

$$\alpha \cdot \beta = 00001,\ \alpha \oplus \beta = 10110.$$

Отметим, что операция исключающего сложения обладает свойством обратимости, то есть если к последовательности α прибавить β дважды, то мы снова получим α:

$$(\alpha \oplus \beta) \oplus \beta = \alpha.$$

Задача. *Упростите выражение* $\alpha(\alpha \oplus \beta) \oplus \beta\alpha$.

ШИФР ВЕРНАМА

Пусть у нас α означает исходную двоичную последовательность, k – другую последовательность той же длины, которая служит ключом. Через β будем обозначать шифртекст – последовательность, полученную после шифрования исходного текста α.

Схема, в которой шифрование задаётся формулой $\beta = \alpha \oplus k$, – это шифр Вернама (1917 г.), на котором основываются поточные шифры.

Например, последовательность $\alpha = 10001$, зашифрованная с использованием ключа $k = 00111$, примет вид $\beta = 10110$.

Расшифровать текст можно, ещё раз прибавив ключ: $\alpha = \beta \oplus k$.

Другое название ключа в шифре Вернама – *одноразовый шифр-блокнот* (*one time pad*, иногда переводят «одноразовая лента». Гильберт Вернам использовал бумажные ленты для исходного текста и для ключа в своём телеграфном аппарате для автоматизации шифрования).

Ключ действительно можно использовать только раз, так как при сложении двух шифртекстов, соответствующих одному

ключу, получается сумма исходных текстов, что дает много информации об исходных текстах и даже часто позволяет их прочесть.

Клод Шеннон доказал в 1949 году, что при совершенно случайном ключе, используемом один раз, шифр Вернама является совершенно секретной (в другой терминологии – абсолютно стойкой) криптосистемой, то есть перехват шифртекста не дает никакой информации о переданном сообщении. Это единственный в настоящее время шифр с таким свойством. Он использовался для обмена сообщениями между руководителями СССР и США во времена холодной войны и некоторыми секретными службами.

Следует обратить внимание на то, что теперь необходимо найти способ передать получателю секретный ключ такой же длины, как и исходное сообщение. Задача пересылки такого ключа по каналам связи с сохранением секретности столь же сложна, как и пересылка самого сообщения. Поэтому ключ передают специальные курьеры, и это стоит так дорого, что используется лишь в особых случаях.

В связи с этим чаще всего отправитель и получатель сообщений выбирают вместо ключа в шифре Вернама *псевдослучайную последовательность*. На первый взгляд кажется, что нули и единицы, составляющие эту последовательность, расставлены хаотично, однако на самом деле такая последовательность выводится по специальным правилам (не секретным) из короткого секретного ключа. Такая последовательность носит название *ключевой поток* или *гамма* (*key stream*, *gamma*), и для её обозначения будем использовать букву γ.

Теперь процессы выглядят так:

– отправитель по короткому ключу k строит последовательность γ и шифрует $\beta = \alpha \oplus \gamma$,

– получатель по тому же ключу k строит последовательность γи расшифровывает $\alpha = \beta \oplus \gamma$.

Пример. Пусть выбран ключ $k = 0011$. Правило построения гаммы возьмём такое: первые 4 бита гаммы совпадают с k, каждые следующие 4 получаются циклическим сдвигом предыдущей четверки на один бит влево.

Получается ключевой поток
$\gamma = 0011\ 0110\ 1100\ 1001\ 0011\ \ldots$

Таким способом получается сколь угодно длинная последовательность. Сразу скажем, что приведённый способ не рекомендуется для построения гамм, он слишком легко вскрывается.

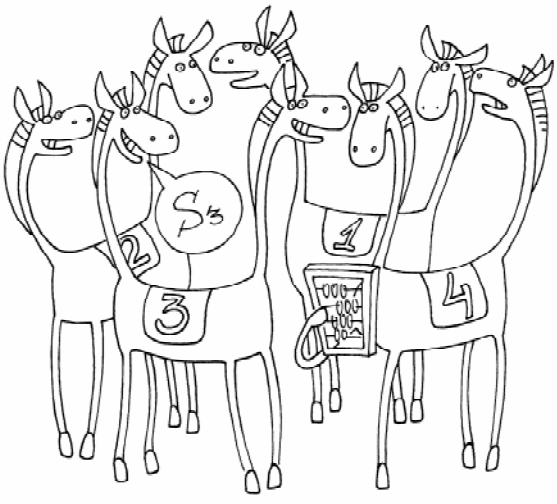
ЛИНЕЙНЫЕ РЕГИСТРЫ СДВИГА

Распространённый способ порождать длинную (вообще говоря, бесконечную) последовательность на основе имеющейся короткой – использование регистров сдвига с линейной обратной связью, для которых принято сокращённое обозначение LFSR (Linear Feedback Shift Register).

Схематично LFSR представляется устройством, имеющим L ячеек, где L – длина регистра и она же – длина секретного ключа. Ячейка вмещает один бит, и изначально в ячейках помещается некоторый набор битов $S^{(0)} = s_0, s_1, \ldots, s_{L-1}$, называемый начальным состоянием LFSR.

s_0	s_1	s_2	…	s_{L-1}
L	L–1	L–2	…	1

Снизу написаны номера ячеек. Мы на схеме пронумеровали их от 1 до L справа



Распространённый способ порождать длинную... последовательность на основе имеющейся короткой — использование регистров сдвига с линейной обратной связью...

налево, как это часто делается для удобства дальнейшего описания.

Работа LFSR проводится тактами, то есть шаг за шагом. На каждом такте i при $i = 1, 2, \ldots$ на выход подаётся бит из левой ячейки, остальные смещаются на одну ячейку левее, а в освободившуюся правую ячейку помещается новое значение. Как оно вычисляется, увидим ниже. На выходе бит за битом образуется последовательность, начинающаяся с s_0. Процесс можно изобразить так (см. табл. 1).

Состояние регистра после i-го такта обозначим $S^{(i)}$. Рассмотрим подробнее, как LFSR переходит от состояния $S^{(i-1)} = s_{i-1}, s_i, \ldots, s_{i+L-2}$ к новому состоянию $S^{(i)} = s_i, s_{i+1}, \ldots, s_{i+L-1}$, то есть как вычисляется новый бит s_{i+L-1}.

Некоторые ячейки – какие именно, определяется устройством конкретного регистра, – являются *отводами*. Это значит, что значения этих ячеек идут на сумматор, то есть складываются между собой по правилу исключающего сложения. И новый бит – это получающаяся сумма.

Пусть, например, в LFSR длины $L = 4$ начальное состояние определено как $S^{(0)} = 1010$. Поместим эти значения в ячейки регистра. Пусть отводами здесь являются 3-я и 4-я ячейки справа, содержащие в начальный момент s_1 и s_0:

$s_4 = 0 \oplus 1 = 1$			
↑			
$\oplus$			
↑	↑		
$s_0 = 1$	$s_1 = 0$	$s_2 = 1$	$s_3 = 0$
4	3	2	1

В результате суммирования получается следующий бит s_4.

Продолжая вычисления, получаем в данном примере табл. 2.

В правом столбце таблицы возникает последовательность, первые L битов которой определяются начальным состоянием регистра, а последующие определяются устройством регистра (отводами). Такую последовательность $s_0 s_1 \ldots s_{n-1}$ при любом числе элементов $n \geq L$ будем называть последовательностью, *порождаемой* (или *генерируемой*) данным LFSR.

Как мы видим, в этом примере $S^{(15)} = S^{(0)}$, и затем в генерируемой последовательности биты начнут повторяться, что вполне объяснимо. Действительно, ненулевых возможных состояний регистра длины L существует ровно $2^L - 1$, это число различных последовательностей длины L, состоящих из нулей и единиц, за вычетом последовательности из одних нулей. (Подумайте, почему именно такое количество). Значит, самое большее через $2^L - 1$ шаг мы придём к уже встречавшемуся состоянию. Нулевое начальное состояние не рассматривают, потому что тогда LFSR при любых отводах будет выдавать последовательность из одних нулей.

Таким образом, последовательность, генерируемая LFSR, будет, начиная с какого-то шага, периодической с периодом не более $2^L - 1$. Если при этом, как в нашем примере, L-я (самая левая) ячейка является отводом, то последовательность будет периодической с самого начала.

Так как мы желаем, чтобы эта последовательность как можно больше походила на случайную, то период хотелось бы иметь как можно длиннее. В нашем примере пе-

Табл. 1

Такт	Выходная последовательность	Состояние регистра в конце такта				
1	$s_0 \leftarrow$	s_1	s_2	…	s_{L-1}	s_L
2	$s_0, s_1 \leftarrow$	s_2	s_3	…	s_L	s_{L+1}
…	…	…				
$i-1$	$s_0, s_1, \ldots, s_{i-2} \leftarrow$	s_{i-1}	s_i	…	s_{i+L-3}	s_{i+L-2}
i	$s_0, s_1, \ldots, s_{i-2}, s_{i-1} \leftarrow$	s_i	s_{i+1}	…	s_{i+L-2}	s_{i+L-1}
$i+1$	$s_0, s_1, \ldots, s_{i-2}, s_{i-1}, s_i \leftarrow$	s_{i+1}	s_{i+2}	…	s_{i+L-1}	s_{i+L}
…	…	…				

Табл. 2

шаг i	s_{L-i-1}, найден по $S^{(i-1)}$	состояние $S^{(i)}$	на выход
0		$s_0\, s_1\, s_2\, s_3 = 1010$	
1	$s_4 = 1$	$s_1\, s_2\, s_3\, s_4 = 0101$	$s_0 = 1$
2	$s_5 = 1$	$s_2\, s_3\, s_4\, s_5 = 1011$	$s_1 = 0$
3	$s_6 = 1$	$s_3\, s_4\, s_5\, s_6 = 0111$	$s_2 = 1$
4	$s_7 = 1$	$s_4\, s_5\, s_6\, s_7 = 1111$	$s_3 = 0$
5	$s_8 = 0$	$s_5\, s_6\, s_7\, s_8 = 1110$	$s_4 = 1$
6	$s_9 = 0$	$s_6\, s_7\, s_8\, s_9 = 1100$	$s_5 = 1$
7	$s_{10} = 0$	$s_7\, s_8\, s_9\, s_{10} = 1000$	$s_6 = 1$
8	$s_{11} = 1$	$s_8\, s_9\, s_{10}\, s_{11} = 0001$	$s_7 = 1$
9	$s_{12} = 0$	$s_9\, s_{10}\, s_{11}\, s_{12} = 0010$	$s_8 = 0$
10	$s_{13} = 0$	$s_{10}\, s_{11}\, s_{12}\, s_{13} = 0100$	$s_9 = 0$
11	$s_{14} = 1$	$s_{11}\, s_{12}\, s_{13}\, s_{14} = 1001$	$s_{10} = 0$
12	$s_{15} = 1$	$s_{12}\, s_{13}\, s_{14}\, s_{15} = 0011$	$s_{11} = 1$
13	$s_{16} = 0$	$s_{13}\, s_{14}\, s_{15}\, s_{16} = 0110$	$s_{12} = 0$
14	$s_{17} = 1$	$s_{14}\, s_{15}\, s_{16}\, s_{17} = 1101$	$s_{13} = 0$
15	$s_{18} = 0$	$s_{15}\, s_{16}\, s_{17}\, s_{18} = 1010$	$s_{14} = 1$
…			

риод равен $15 = 2^4 - 1$, то есть является максимально возможным для LFSR длины 4.

Для анализа свойства выходной последовательности LFSR, вводят битовые значения c_i, $i = 1, 2, \ldots, L$, такие, что $c_i = 1$, если i-я ячейка (нумерация справа) является отводом, и $c_i = 0$ в противном случае.

В рассмотренном выше LFSR отводами были ячейки 3 и 4, поэтому для него

$c_1 = c_2 = 0$,

$c_3 = c_4 = 1$.

Значения c_i, равные 1, также называются отводами.

Правило вычисления нового бита s_{i+L-1} по состоянию $S^{(i)}$ тогда запишется так:

$$s_{i+L-1} = c_1 s_{i+L-2} \oplus c_2 s_{i+L-3} \oplus \ldots \oplus c_{L-1} s_i \oplus c_L s_{i-1},\ i = 1, 2, 3, \ldots \qquad (1)$$

Переобозначая $i := i + L - 1$, запишем (1) в виде правила для вычисления i–го бита:

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

Это похоже на то, что мы линейку

c_1	c_2	…	c_L

совмещаем с линейкой

s_{i-1}	s_{i-2}	…	s_{i-L}

и перемножаем соответствующие элементы, а затем складываем результаты.

Если перехваченная последовательность состоит не менее чем из $2L$ битов, то при известной длине регистра L легко восстановить набор коэффициентов c_i, используемый данным регистром. Для этого нужно решить систему из L линейных уравнений:

$$\begin{cases} s_{L-1}c_1 \oplus s_{L-2}c_2 \oplus \ldots \oplus s_1 c_{L-1} \oplus s_0 c_L = s_L \\ s_L c_1 \oplus s_{L-1}c_2 \oplus \ldots \oplus s_2 c_{L-1} \oplus s_1 c_L = s_{L+1} \\ \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \\ s_{2L-3}c_1 \oplus s_{2L-4}c_2 \oplus \ldots \oplus s_{L-1}c_{L-1} \oplus s_{L-2} \cdot c_L = s_{2L-2} \\ s_{2L-2}c_1 \oplus s_{2L-3}c_2 \oplus \ldots \oplus s_L c_{L-1} \oplus s_{L-1}c_L = s_{2L-1} \end{cases}$$

Пример. Пусть перехвачена последовательность 01011110 и известно, что $L = 4$.

Запишем уравнения (2) для $i = 4, 5, 6, 7$.

$$\begin{cases} 1 \cdot c_1 \oplus 0 \cdot c_2 \oplus 1 \cdot c_3 \oplus 0 \cdot c_4 = 1 \\ 1 \cdot c_1 \oplus 1 \cdot c_2 \oplus 0 \cdot c_3 \oplus 1 \cdot c_4 = 1 \\ 1 \cdot c_1 \oplus 1 \cdot c_2 \oplus 1 \cdot c_3 \oplus 0 \cdot c_4 = 1 \\ 1 \cdot c_1 \oplus 1 \cdot c_2 \oplus 1 \cdot c_3 \oplus 1 \cdot c_4 = 0 \end{cases}$$

Решим систему методом последовательного исключения неизвестных. Для этого прибавим сначала первое уравнение ко всем остальным уравнениям системы, исключая тем самым c_1 из всех уравнений, кроме первого. Получим

$$\begin{cases} 1\cdot c_1 \oplus 0\cdot c_2 \oplus 1\cdot c_3 \oplus 0\cdot c_4 = 1 \\ 1\cdot c_2 \oplus 1\cdot c_3 \oplus 1\cdot c_4 = 0 \\ 1\cdot c_2 \oplus 0\cdot c_3 \oplus 0\cdot c_4 = 0 \\ 1\cdot c_2 \oplus 0\cdot c_3 \oplus 1\cdot c_4 = 1 \end{cases}$$

Теперь добавим 2-ое уравнение системы к 3-ему и 4-ому уравнениям, исключая из этих уравнений c_2, а затем 3-е уравнение к 4-ому, исключая c_3. Окончательно имеем

$$\begin{cases} 1\cdot c_1 \oplus 0\cdot c_2 \oplus 1\cdot c_3 \oplus 0\cdot c_4 = 1 \\ 1\cdot c_2 \oplus 1\cdot c_3 \oplus 1\cdot c_4 = 0 \\ 1\cdot c_3 \oplus 1\cdot c_4 = 0 \\ 1\cdot c_4 = 1 \end{cases}$$

или

$$\begin{cases} c_1 \oplus c_3 = 1 \\ c_2 \oplus c_3 \oplus c_4 = 0 \\ c_3 \oplus c_4 = 0 \\ c_4 = 1 \end{cases}$$

Используя определение логического исключающего сложения, получаем, двигаясь от нижнего уравнения к верхнему, $c_4 = 1$, $c_3 = 1$, $c_2 = 0$, $c_1 = 0$. Итак, набор коэффициентов восстановлен: 0011.

Задача. *По набору битов* 1000011011 *восстановить отводы порождающего ее регистра длины* $L = 5$.

Предлагаем читателю составить две компьютерные программы. Первая вычисляет по заданным длине регистра L, отводам и начальному состоянию выходную последовательность произвольной длины (не меньше $2L$). Вторая по заданной выходной последовательности при известном значении L восстанавливает коэффициенты $c_1, c_2, \ldots, c_L$.

Если имеется последовательность битов, порождённая регистром неизвестной длины, то оказывается, что задача восстановления отводов также разрешима.

Литература

1. *Х.К.А. ван Тилборг.* Основы криптологии. М.: Мир, 2006.
2. *Н. Смарт.* Криптография. М.: Техносфера, 2005.

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.