

ЭФФЕКТИВНЫЙ МЕТОД РАНДОМИЗАЦИИ СООБЩЕНИЙ НА ОСНОВЕ АРИФМЕТИЧЕСКОГО КОДИРОВАНИЯ*)

А. Н. Фионов

Рассматривается задача полной рандомизации сообщений, возникающая в криптографии при конструировании безусловно стойких шифров с секретным ключом [6, 7]. Одним из основных параметров любого метода рандомизации является избыточность r , определяемая как разность между средней длиной кодового слова и энтропией на символ источника. Для получения произвольно низкой избыточности описанным в [3, 11] методом требуется $O(1/r)$ памяти и $O(\log^2(1/r) \log \log(1/r))$ времени кодирования и декодирования. В настоящей работе предлагается метод, для которого объем памяти и время определяются соответственно как $O(\log(1/r))$ и $O(\log(1/r) \log \log(1/r) \log \log \log(1/r))$ при $r \rightarrow 0$. Предлагаемый метод, однако, использует существенно большее количество случайных символов, чем известные методы.

Введение

Центральной проблемой криптографии является проблема стойкости криптосистем. Один из радикальных путей ее решения — построение безусловно стойких (невскрываемых) шифров. Понятие безусловной стойкости было введено К. Шенноном [4] и означает такое свойство шифра, при котором никакая сколь угодно длинная последовательность шифротекста не увеличивает априорных знаний об использованном секретном ключе (в теоретико-информационном смысле), что, в свою очередь, возможно, если только шифротекст статистически не зависит от ключа. В работе [7] было показано, что если сообщение источника (открытый текст) закодировать таким образом, чтобы кодовая последовательность была полностью случайной, т. е. состоящей из равновероятных и независимых кодовых символов, то зашифрованная последовательность (при определенных ограничениях, накладываемых

*) Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (код проекта 96-01-00052).

на шифр) также становится полностью случайной и независимой от ключа. Отсюда следует безусловная стойкость криптосистемы. В этом случае криптоаналитик не получает никакой возможности раскрыть зашифрованное сообщение, кроме простого угадывания ключа, и при достаточно большой мощности множества ключей шифр становится практически невскрываемым.

Задача преобразования текста в последовательность независимых и равновероятных символов может быть решена путем рандомизации (случайного кодирования) сообщений источника. Идея рандомизации была известна еще Гауссу (см., например, [8]) и использовалась им в криптографических целях. Чтобы пояснить ее суть, приведем пример из [6]. Допустим, что бернуллиевский источник порождает буквы из алфавита $A = \{a, b\}$ с вероятностями $P(a) = 3/4$, $P(b) = 1/4$. Закодируем каждый символ сообщения источника в соответствии с табл. 1.

Табл. 1. Омофонное кодирование
равномерным кодом ($P(a) = 3/4$, $P(b) = 1/4$)

Символ	Кодовое слово	Вероятность выбора
a	00	1/3
	01	1/3
	10	1/3
b	11	1

Здесь символу a соответствуют три различных кодовых слова, выбираемых при кодировании случайным образом. Можно считать, что символ a представляется тремя новыми символами, называемыми *омофонами*, которые затем кодируются кодовыми словами 00, 01 и 10 (термин «омофон» заимствован из лингвистики, в которой он обозначает слова, имеющие одинаковое произношение, но различное написание). Отсюда происходит название целого класса методов — омофонное кодирование.

Легко убедиться в том, что результирующая кодовая последовательность является полностью случайной, т. е. все вхождения нулей и единиц равновероятны и независимы. Описанная схема имеет два основных недостатка: она не работает при произвольных рациональных вероятностях символов источника и приводит к сильному «растяжению» сообщения, так как длина кодового слова определяется логарифмом минимальной вероятности символа. Степень «растяжения» мы будем характеризовать избыточностью. Под избыточностью r метода рандомизации понимается разность между средней длиной кодового слова и энтропией

источника. Для приведенного примера энтропия на букву источника приблизительно равна 0,81 бит. Следовательно, избыточность составляет примерно $2 - 0,81 = 1,19$ бит на символ сообщения.

К. Гюнтер [6] в 1988 г. предложил метод омофонного кодирования с переменной длиной кодового слова. В этом методе кодовая таблица для приведенного выше случая имеет следующий вид:

Табл. 2. Омофонное кодирование
неравномерным кодом ($P(a) = 3/4$, $P(b) = 1/4$)

Символ	Кодовое слово	Вероятность выбора
a	0	2/3
	10	1/3
b	11	1

По сравнению с табл. 1 два кодовых слова 00 и 01, представляющие символ a , здесь заменены на одно кодовое слово 0 с удвоенной вероятностью выбора. Легко показать, что результирующая кодовая последовательность также является полностью случайной при очевидном сокращении средней длины кодового слова. Для рассматриваемого примера средняя длина кодового слова составляет 1,5 бита и избыточность приблизительно равна $1,5 - 0,81 = 0,69$ бит на символ сообщения. Метод Гюнтера позволяет также строить код при произвольных рациональных вероятностях символов источника.

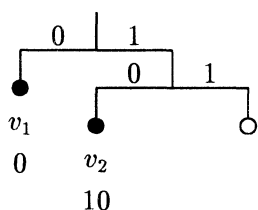


Рис. 1. Дерево выбора
омофонов символа a

В работе [7] дан теоретико-информационный анализ омофонного кодирования и предложен метод реализации вероятностного выбора омофонов при помощи случайных (равновероятных и независимых) бит, поступающих от дополнительного двоичного источника.

Для рассмотренного примера реализация вероятностного выбора омофонов при кодировании символа a возможна с помощью двоичного дерева (рис. 1).

Символ a представляется омофонами v_1 и v_2 , которые кодируются соответственно как 0 и 10. Листья дерева являются либо омофонами (черный кружок), либо остаются пустыми (белый кружок). Дерево просматривается, начиная с корня, в соответствии со значениями случайных бит, порождаемых дополнительным источником. Если переход

осуществляется в черный кружок, то производится выбор соответствующего омофона; если переход происходит в белый кружок, то проход по дереву необходимо повторить с использованием новых случайных бит. Легко видеть, что вероятность выбора v_1 в два раза превышает вероятность выбора v_2 . Это означает, что вероятности выбора $P(v_1)$ и $P(v_2)$ равны соответственно $2/3$ и $1/3$, что и требуется кодовой таблицей. Можно подсчитать, что при выборе омофона для символа a в среднем понадобится 2 случайных бита.

В работе [7] было показано, что существует оптимальная схема омофонного кодирования, дающая минимальную избыточность. В случае ее использования посимвольное кодирование сообщения источника может быть выполнено с избыточностью, не превышающей 2 бит, и расходом в среднем не более 4 случайных бит от дополнительного источника на каждый символ сообщения. Во многих случаях, особенно для источников с низкой энтропией, избыточность 2 бита на символ сообщения может оказаться слишком высокой. Классическим путем уменьшения избыточности является кодирование блоков символов. Однако традиционные методы блочного кодирования оказываются практически нереализуемыми из-за экспоненциального роста объема кодовой таблицы при увеличении размера блока.

В работах [3, 11] был предложен метод омофонного кодирования блоков символов, требующий лишь линейного роста объема памяти с увеличением размера блока. Лежащая в его основе схема построения омофонного кода будет рассмотрена в следующем разделе при описании нового метода. Здесь же отметим, что эта схема используется совместно с методом эффективного блочного кодирования, предложенного в работах [2, 10]. Этот метод позволяет получать заданную произвольно низкую избыточность на символ сообщения и имеет асимптотические оценки объема памяти кодера (декодера) и времени кодирования (декодирования) соответственно $O(1/r)$ и $O(\log^2 1/r \log \log 1/r)$ при $r \rightarrow 0$. Такие оценки объясняются необходимостью хранения в памяти всего кодового слова для блока символов и выполнения арифметических операций с возрастающей точностью, определяемой размером блока.

В настоящей работе предлагается иной подход, основная идея которого состоит в использовании арифметического кодирования при сохранении постоянной точности вычислений. Арифметическое кодирование было открыто в конце 70-х годов и широко исследовано в середине 80-х (см., например, [5, 9]). При посимвольном кодировании сообщения источника этим методом ограничение на то, что символы сообщения должны кодироваться целым числом бит, становится несущественным. Фактически каждый символ добавляет в кодовую последовательность

дробное число бит, определяемое той информацией, которую несет символ, что приводит к практически безызбыточному кодированию. Для решения задачи полной рандомизации в настоящей работе арифметическое кодирование дополнено процедурой выделения и выбора омофонов для каждого кодируемого символа (прием, названный нами «разделением интервала»). Новый метод, как и метод из [3, 11], позволяет получать заданную избыточность, но при $r \rightarrow 0$ объем памяти и время кодирования и декодирования растут как $O(\log(1/r))$ и $O(\log(1/r) \log \log(1/r) \log \log \log(1/r))$ соответственно.

Предлагаемый метод, в свою очередь, уступает методу из [3, 11] по количеству расходуемых случайных бит, что связано с необходимостью реализации вероятностного выбора омофонов для каждого кодируемого символа. Если для метода из [3, 11] число N_r случайных бит на символ сообщения (как функция избыточности r) удовлетворяет неравенству $N_r < 4r$, то для предлагаемого метода N_r приближенно равно $2 + r$, что существенно хуже при малых значениях r .

Остановимся на содержании работы. В разд. 1 основная идея предлагаемого метода рассматривается на примере построения кода для конкретного сообщения источника. Затем в разд. 2 приводится точное описание метода и устанавливаются его основные свойства. В разд. 3 описывается метод вероятностного выбора, который может быть использован при разделении интервала. Наконец, в приложении даются алгоритмы омофонного кодирования интервала, заимствованные (с некоторыми изменениями) из [3, 11].

1. Основная идея метода

Рассмотрим следующий пример. Пусть дан бернуллиевский источник, порождающий буквы из алфавита $A = \{a, b, c\}$ с вероятностями

$$P(a) = 7/16, P(b) = 6/16, P(c) = 3/16. \quad (1)$$

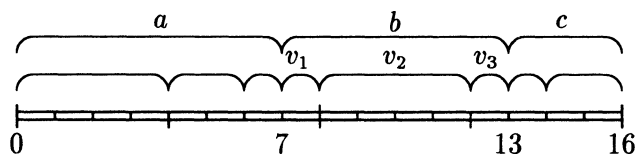


Рис. 2. Распределение букв с вероятностями из (1)

Распределим буквы алфавита на отрезке $[0, 16)$ так, как показано на рис. 2. Каждая буква представлена непрерывным интервалом, длина которого пропорциональна вероятности появления буквы ($[0, 7)$ для a , $[7, 13)$ для b и $[13, 16)$ для c).

Разделим полный интервал $[0, 16)$ на две равные части, каждую часть вновь разделим пополам и будем продолжать этот процесс до тех пор, пока каждый получающийся отрезок не войдет полностью в тот или иной интервал. В результате получим разбиение интервалов на подынтервалы, что соответствует представлению символов алфавита источника совокупностью омофонов (рис. 2).

Если в процессе деления левым частям будем приписывать ноль, а правым — единицу, то получим код каждого подынтервала. После этого достаточно будет выбрать тот или иной подынтервал с вероятностью, пропорциональной его длине. Результаты описанного процесса приведены в табл. 3.

Табл. 3. Омофонное кодирование символов при распределении вероятностей из (1)

Символ	Кодовое слово	Вероятность выбора
<i>a</i>	00	4/7
	010	2/7
	0110	1/7
<i>b</i>	0111	1/6
	10	4/6
	1100	1/6
<i>c</i>	1101	1/3
	111	2/3

В приведенном примере средняя длина кодового слова равна 2,75 бит при энтропии источника приблизительно 1,5 бита, что дает избыточность примерно 1,25 бита на символ сообщения. Данная схема была предложена в [3, 11] и используется в качестве основы для метода, предлагаемого в настоящей работе.

На рис. 3 изображены два дерева для выбора омофонов символа $u = b$ (см. табл. 3). Легко подсчитать, что для выбора омофона по первому дереву требуется в среднем $2\frac{1}{3}$ случайного бита, а по второму дереву — $7\frac{1}{3}$ бита. Второе дерево, однако, имеет то преимущество, что для выбора омофонов можно обойтись без явного построения дерева, так как коды омофонов определяются последовательностью приведших к их выбору случайных бит, а условие попадания в белый кружок может быть проверено путем сравнения получающегося кода с границами интервала для кодируемого символа. Именно второе дерево будет использоваться в дальнейшем при омофонном кодировании интервала.

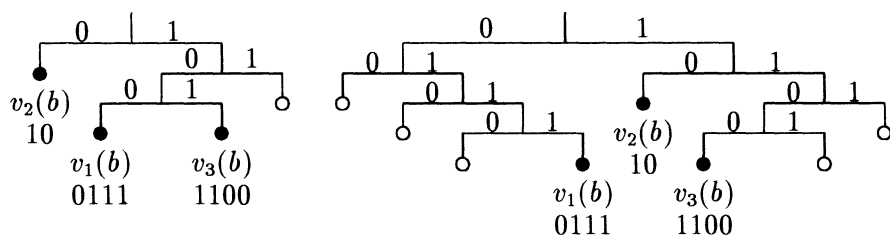


Рис. 3. Деревья для выбора омофонов символа *b*

Итак, при посимвольном омофонном кодировании для каждого символа сообщения выбирается соответствующий ему интервал и производится омофонное кодирование этого интервала. Основная идея арифметического кодирования (см., например, [5, 9]) заключается в том, что кодирование интервала на каждом шаге не производится. Вместо этого на интервале, соответствующем первому символу сообщения, рассматривается новое распределение символов алфавита источника, в котором выбирается интервал, соответствующий второму символу сообщения, и т. д. Иными словами, каждый последующий символ сообщения сужает текущий интервал до интервала, соответствующего этому символу. В результате получается интервал, соответствующий всему сообщению. Для рандомизации сообщения необходимо произвести омофонное кодирование этого заключительного интервала.

На пути реализации описанной идеи возникает следующая трудность: разрядность чисел, представляющих границы интервала, а следовательно, и точность вычислений, экспоненциально возрастает при увеличении длины сообщения. Например, для представления интервалов для символов с вероятностями из (1) достаточно разделить отрезок на 16 частей (см. рис. 2), что соответствует представлению границ интервалов в виде 4-значных двоичных чисел. Для представления интервалов для двухбуквенных сообщений отрезок нужно разделить уже на $16^2 = 256$ частей, что соответствует представлению границ интервалов в виде 8-значных двоичных чисел. При арифметическом кодировании постоянная точность вычислений сохраняется за счет масштабирования и округления интервалов. В случае сжатия данных округление интервала лишь увеличивает избыточность кода, но при решении задачи рандомизации оно недопустимо. Вместо этого мы делим интервал на части (или подынтервалы) и выбираем одну из частей с вероятностью, пропорциональной ее длине. Это соответствует представлению кодируемого символа омофонами.

В качестве примера рассмотрим кодирование сообщения abc . Для представления границ интервала будем использовать 4-значные

двоичные числа, записываемые для удобства в десятичной системе счисления (как будет показано в разд. 2, точность представления границ интервала зависит от минимальной вероятности символа и заданной избыточности). Обозначим через $|a|$ длину интервала для символа a , через $|ab|$ — суммарную длину интервалов для символов a и b . Процесс построения кодового слова изображен на рис. 4.

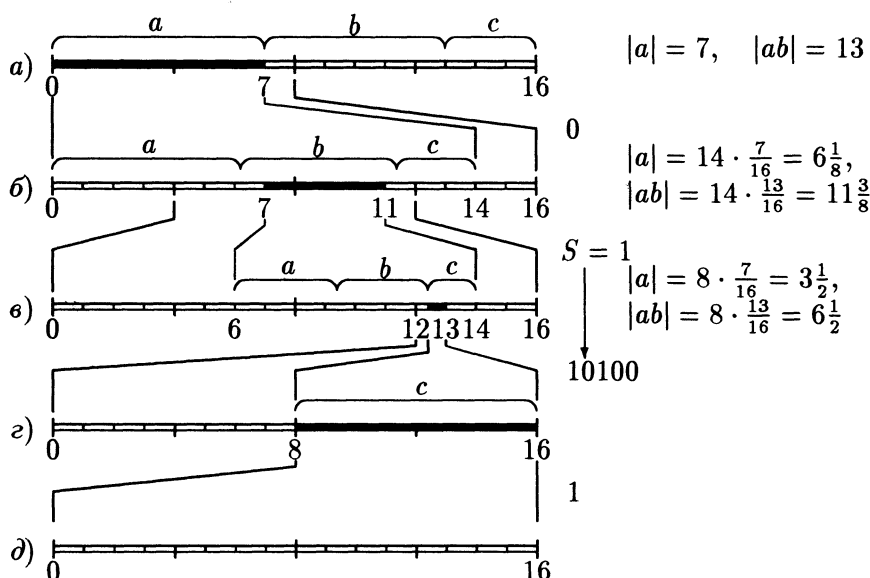


Рис. 4. Кодирование сообщения abc

Получив первый символ сообщения, выбираем интервал $[0, 7]$, соответствующий букве a (см. рис. 4, а). Заключительный интервал для сообщения abc будет находиться внутри интервала $[0, 7]$, а интервал $[0, 7]$ целиком лежит в левой половине полного интервала. Поэтому первый бит кода заключительного интервала будет равен нулю. Передадим ноль на выход кодера и исключим из рассмотрения правую половину полного интервала. Увеличим масштаб оставшейся (левой) половины вдвое. При этом интервал $[0, 7]$ перейдет в интервал $[0, 14]$ (см. рис. 4, б). Описанный процесс, в ходе которого был получен кодовый бит, назовем S -масштабированием.

Найдем распределение букв алфавита источника на интервале $[0, 14]$. Для этого вычислим $|a| = dP(a)$, $|ab| = d(P(a) + P(b))$, где d есть длина интервала (см. рис. 4, б). Для следующего символа сообщения выберем интервал $[6\frac{1}{8}, 11\frac{3}{8}]$, соответствующий букве b . Так как границы интервала не являются целыми числами, разделим интервал на три части с длинами $7/8$, 4 и $3/8$ и произведем вероятностный

выбор одной из них. Допустим, выбрана средняя часть длиной 4, т. е. интервал $[7, 11)$. Попытаемся его масштабировать. Интервал $[7, 11)$ не принадлежит какой-либо одной половине полного интервала. Поэтому его S -масштабирование невозможно. Однако он целиком лежит во второй и третьей четвертях полного интервала. Это означает, что следующие два кодовых бита могут быть либо 01, либо 10. Важно, что второй бит является обратным по отношению к первому. Введем переменную S и запишем $S = 1$, чтобы запомнить, что после очередного кодового бита (когда он определится) нужно будет вставить один обратный бит. Увеличим вдвое масштаб второй и третьей четвертей. При этом интервал $[7, 11)$ перейдет в интервал $[6, 14)$, а середина полного интервала останется на месте (см. рис. 4, в). Описанный процесс, в ходе которого формируется значение S , назовем S -масштабированием.

Найдем распределение букв алфавита источника на интервале $[6, 14)$ (см. рис. 4, в). Для последнего символа сообщения выберем интервал $[12\frac{1}{2}, 14)$, соответствующий букве c . Разделим интервал на два подынтервала $[12\frac{1}{2}, 13)$ и $[13, 14)$. Допустим, теперь выбран первый из них. Он лежит внутри интервала $[12, 13)$ с целочисленными границами. Сначала масштабируем интервал $[12, 13)$. Для этого нужно проделать четыре S -масштабирования, в ходе которых будут получены четыре кодовых бита 1100 (двоичное представление числа 12). После первого из них, чтобы учесть S -масштабирование на предыдущем шаге, нужно вставить ноль. В результате масштабирования интервал $[12\frac{1}{2}, 13)$ перейдет в интервал $[8, 16)$ (см. рис. 4, г). Еще раз масштабируем его с передачей на выход кодера единицы. Интервал $[8, 16)$ переходит в интервал $[0, 16)$, который является заключительным интервалом для сообщения abc . Теперь необходимо провести омофонное кодирование заключительного интервала. В рассмотренном примере омофонное кодирование вырождается в пустую операцию, так как заключительный интервал совпадает с полным интервалом.

В результате мы имеем кодовое слово

$$C_1(abc) = 0101001.$$

Вероятность появления на выходе кодера кодового слова C_1 можно вычислить следующим образом:

$$P(C_1) = P(abc)P(C_1|abc),$$

где первый множитель есть вероятность появления сообщения abc , а второй — условная вероятность того, что из всех возможных кодовых слов для сообщения abc выбрано именно C_1 . Эта условная вероятность определяется произведением вероятностей выбора частей при

разделении интервалов. Для рассмотренного примера

$$P(C_1) = \left(\frac{7}{16} \cdot \frac{6}{16} \cdot \frac{3}{16} \right) \left(\frac{4}{7/8 + 4 + 3/8} \cdot \frac{1/2}{1/2 + 1} \right) = \frac{1}{128} = \left(\frac{1}{2} \right)^7,$$

что совпадает с вероятностью появления последовательности из семи равновероятных и независимых бит.

Собственная информация сообщения abc равна $-\log P(abc) \approx 5$ бит, так что избыточность составляет $2/3$ бита на символ. Если бы при кодировании была выбрана правая часть интервала для буквы c , то мы имели бы $C_2(abc) = 010101$ и избыточность составила бы $1/3$ бита на символ сообщения. В следующем разделе показано, что дальнейшего уменьшения избыточности можно достичь путем увеличения точности представления границ интервала.

2. Описание метода и его свойства

Обозначим через $U = u_1 u_2 u_3 \dots$ произвольную последовательность символов в алфавите $A = \{a_1, a_2, \dots, a_N\}$, порождаемую некоторым источником информации. Нас не будет интересовать природа источника информации. Мы будем кодировать последовательность U посимвольно и потребуем только, чтобы на момент кодирования некоторого символа u из U нам было известно распределение вероятностей $P_u(a_1), P_u(a_2), \dots, P_u(a_N)$ появления букв алфавита A . Вообще говоря, нас будут интересовать кумулятивные вероятности $Q_u(a_1), \dots, Q_u(a_N)$, определяемые следующим образом:

$$Q_u(a_1) = 0, \quad Q_u(a_i) = \sum_{j < i} P_u(a_j), \quad i = 2, 3, \dots, N.$$

Введем также вспомогательную величину $\hat{Q}_u(u) = Q_u(u) + P_u(u)$. В дальнейшем для упрощения обозначений будем опускать индекс u , помня о том, что для каждого символа сообщения U распределение вероятностей может быть различным, т. е. сам метод кодирования и декодирования не накладывает в этом отношении никаких ограничений.

Последовательность U может быть как конечной, так и бесконечной. В любом случае, чтобы избежать некоторых затруднений, объясняемых ниже, последовательность будем разбивать на блоки длины L и кодировать эти блоки отдельно. Для конечных последовательностей в алфавит источника введем специальный символ, завершающий сообщение.

Пусть

$$P(a_1) = \frac{\rho(a_1)}{\delta}, \quad P(a_2) = \frac{\rho(a_2)}{\delta}, \quad \dots, \quad P(a_N) = \frac{\rho(a_N)}{\delta},$$

где ρ и δ — целые числа. Тогда величины $Q(u)$ и $\hat{Q}(u)$ также могут быть представлены как рациональные дроби $Q(u) = \vartheta(u)/\delta$, $\hat{Q}(u) = \hat{\vartheta}(u)/\delta$, причем

$$\begin{aligned} \vartheta(a_1) = 0, \quad \vartheta(a_i) = \hat{\vartheta}(a_{i-1}) = \vartheta(a_{i-1}) + \rho(a_i), \\ i = 2, 3, \dots, N, \quad \hat{\vartheta}(a_N) = \delta. \end{aligned}$$

Из примера в предыдущем разделе (см. рис. 4) видно, что процесс кодирования для каждого символа $u \in U$ состоит из операций вычисления границ интервала, деления интервала и масштабирования. Нам удобно рассмотреть эти операции в обратном порядке.

Обозначим нижнюю и верхнюю границы интервала через l и h , $l < h$. Интервал $[l, h)$ полуоткрытый, так как h является нижней границей для следующего, смежного с ним интервала. Будем считать, что l и h представлены t -битными целыми числами. Для упрощения описания допустим случай $h = 2^t$. Интервал $[0, 2^t)$ будем называть полным интервалом. Во избежание излишнего усложнения метода потребуем, чтобы t удовлетворяло неравенству

$$t \geq 1 - \log P_{\min}(u), \quad (2)$$

где $P_{\min}(u)$ — минимальная для всех распределений ненулевая вероятность символа.

При проведении масштабирования выявим следующие случаи:

- интервал $[l, h)$ целиком лежит в какой-либо половине полного интервала;
- интервал $[l, h)$ целиком лежит во второй и третьей четвертях полного интервала.

В первом случае выполняется C -масштабирование, во втором — S -масштабирование. Если ни тот, ни другой случай не имеет места, интервал масштабировать нельзя. Такой интервал будем называть *нормализованным*.

Чтобы определить попадание интервала $[l, h)$ в какую-либо часть полного интервала, удобно перейти к замкнутому интервалу $[l, g]$, $g = h - 1 + 0,111\dots$, где $0,111\dots$ — бесконечная двоичная дробь. Запишем двоичные представления l и g в следующем виде:

$$\begin{aligned} l &= l_1 l_2 \dots l_t, 000\dots, \\ g &= g_1 g_2 \dots g_t, 111\dots \end{aligned}$$

Для выявления указанных выше двух случаев достаточно сравнить биты чисел l и g . Например, если старшие биты l и g равны нулю ($l_1 = g_1 = 0$), то интервал $[l, g]$ принадлежит левой половине полного

интервала. Если старшие биты l и g равны единице ($l_1 = g_1 = 1$), то интервал $[l, g]$ целиком лежит в правой половине полного интервала. Масштабирование заключается в выдаче на выход кодера соответственно нуля или единицы и сдвиге влево содержимого l и g . При $l_1 \neq g_1$, если $l_2 = 1$ и $g_2 = 0$, интервал $[l, g]$ лежит в пределах второй и третьей четвертей полного интервала. В этом случае масштабирование сводится к записи единицы в специальный счетчик S и сдвиге влево чисел l и g с сохранением старших бит.

В общем случае величины l и g можно представить в виде

$$\begin{aligned} l &= b_1 b_2 \dots b_c 0 11 \dots 1 l_{n+1} \dots l_t, 000 \dots, \\ g &= b_1 b_2 \dots b_c \underbrace{1 00 \dots 0}_s g_{n+1} \dots g_t, 111 \dots, \\ 0 &\leq c \leq t, \quad 0 \leq s \leq t-1, \quad n = c + s + 1. \end{aligned}$$

Тогда в ходе масштабирования на выход кодера передаются биты $b_1 b_2 \dots b_c$, формируется значение счетчика $S = s$ и интервал $[l, g]$ переходит в интервал $[l', g']$, где

$$\begin{aligned} l' &= 0 l_{n+1} \dots l_t 0 \dots 0, 000 \dots, \\ g' &= 1 g_{n+1} \dots g_t 1 \dots 1, 111 \dots \end{aligned}$$

На основании изложенного выше запишем следующий

Алгоритм масштабирования интервала при кодировании

Вход: интервал $[l, h)$ и значение счетчика S , сформированное в предыдущих операциях масштабирования.

$g := h - 1$;

C-масштабирование

пока $l_1 = g_1$

передать на выход кодера бит l_1 ,

S раз передать бит $1 - l_1$, $S := 0$,

$l := 2l, \quad g := 2g + 1 \pmod{2^t}$;

S-масштабирование

пока $l_2 = 1$ и $g_2 = 0$

$S := S + 1$,

$l := 2l, \quad g := 2g + 1 \pmod{2^t}$;

$l_1 := 0, \quad g_1 := 1$;

$h := g + 1$.

Рассмотрим проблему возможного переполнения счетчика S . Это переполнение может произойти, если при кодировании достаточно большого числа символов последовательности U производится только S -масштабирование интервала. Каждый раз при масштабировании значение

S может увеличиться максимум на $t - 1$. Если разрядность счетчика S составляет t_s бит, то условие отсутствия переполнения для последовательности символов длиной L будет $L(t - 1) < 2^{t_s}$. Отсюда получаем ограничение на длину последовательности:

$$L < \frac{2^{t_s}}{t - 1}. \quad (3)$$

Теперь рассмотрим задачу разделения интервала. При кодировании некоторого символа u из U нормализованный интервал $[l, h)$ длиной d сужается до интервала $[l + dQ(u), l + d\hat{Q}(u))$ (рис. 5). При выполнении неравенства (2) вероятности всех символов оказываются не меньше чем $2/2^t$ и соотношение полученного интервала с целочисленной шкалой позволяет разделить его на три части, обозначенные как V^- , V и V^+ .

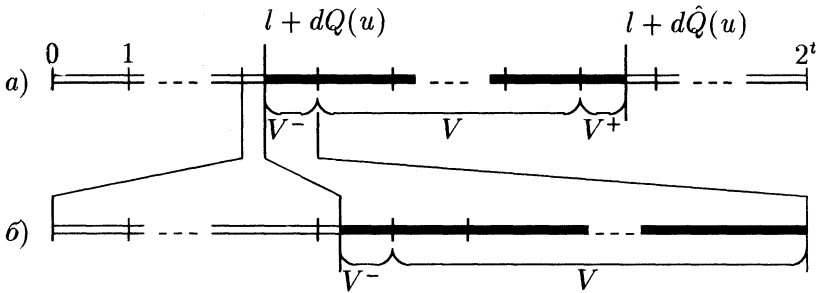


Рис. 5. Разделение интервала

Мы выполним условие полной рандомизации, если для дальнейшего кодирования выберем только одну из этих частей с вероятностью, пропорциональной ее длине.

Из рис. 5 следует, что выбор подынтервала V позволяет остаться в рамках t -битной точности. Если же выбираются подынтервалы V^+ или V^- , то необходимо масштабировать единичный интервал, включающий V^+ или V^- , и повторить процесс разделения. Пример дальнейшего разделения интервала, когда на первой стадии был выбран V^- , показан на рис. 5, б.

Обозначим через I_l и R_l целую часть и остаток, получающиеся в результате вычисления $dQ(u) = d\vartheta(u)/\delta$. Подобным образом обозначим через I_h и R_h целую часть и остаток в операции $d\hat{Q}(u) = d\hat{\vartheta}(u)/\delta$. Тогда длины интервалов V , V^- и V^+ , обозначенные соответственно через v , v^- и v^+ , определяются следующим образом:

$$\begin{aligned} v &= I_h - I_l + \lceil R_l/\delta \rceil, \\ v^- &= \lceil R_l/\delta \rceil - R_l/\delta, \\ v^+ &= R_h/\delta. \end{aligned}$$

Так как для вероятностного выбора этих интервалов важно только соотношение их длин, переведем эти длины в область целых чисел путем умножения на δ . Процедура вероятностного выбора будет рассмотрена в следующем разделе, а теперь дадим

Общий алгоритм кодирования

Инициализация $l := 0, h := 2^t, S := 0$.

Кодирование очередного символа u

(параметры: $\vartheta = \vartheta(u), \hat{\vartheta} = \hat{\vartheta}(u), \delta$)

$d := h - l$,

$I_l := \lfloor d\vartheta/\delta \rfloor, \quad R_l := d\vartheta \pmod{\delta},$

$I_h := \lfloor d\hat{\vartheta}/\delta \rfloor, \quad R_h := d\hat{\vartheta} \pmod{\delta};$

$v^+ := R_h, \quad v^- := \delta - R_l \pmod{\delta}, \quad v := d\hat{\vartheta} - d\vartheta - v^+ - v^-,$

произвести выбор интервала V, V^+ или V^-

с вероятностью, пропорциональной его длине;

если выбран V^+ , то

$l := l + I_h, \quad h := l + 1, \quad \text{масштабировать } [l, h),$

выполнить кодирование с параметрами $\vartheta = 0, \hat{\vartheta} = R_h$;

если выбран V^- , то

$l := l + I_l, \quad h := l + 1, \quad \text{масштабировать } [l, h),$

выполнить кодирование с параметрами $\vartheta = R_l, \hat{\vartheta} = \delta$;

если выбран V , то

$h := l + I_h,$

$l := l + I_l$, если $R_l \neq 0$, то $l := l + 1$,

масштабировать $[l, h)$;

конец кодирования символа u .

Омофонное кодирование заключительного интервала

Вход: нормализованный интервал $[l, h)$.

Если не выполняется условие ($l = 0, h = 2^t$ и $S = 0$), то

определить омофонный код интервала $[l, h)$ $\beta_0\beta_1\ldots\beta_k$,

передать на выход кодера бит β_0 ,

S раз передать бит $1 - \beta_0$,

передать оставшиеся биты $\beta_1\ldots\beta_k$.

Для определения омофонного кода интервала можно использовать метод, предложенный в [3, 11] и проиллюстрированный в примере (см. рис. 2). Точное описание соответствующей процедуры приводится в приложении.

Теперь опишем метод декодирования. Вначале поясним некоторые его детали на примере декодирования первого символа сообщения.

Введем переменную s , в которой будем хранить текущие t бит кодовой последовательности.

Сначала имеем исходный интервал $[l, h)$ длиной d и первые t бит кодовой последовательности в s . Чтобы найти первый закодированный символ, необходимо выяснить, какой из символов алфавита A мог так сузить интервал $[l, h)$, что значение s оказалось внутри этого суженного интервала. Для этого вычислим $I_c = \lfloor (c-l)\delta/d \rfloor$ и $R_c = (c-l)\delta \pmod{d}$ и найдем символ $u \in A$ такой, что выполняется неравенство $\vartheta(u) \leq I_c < \widehat{\vartheta}(u)$.

Если последующие не вошедшие в s биты кодовой последовательности не могут изменить этого неравенства, то u есть закодированный символ. Исследуем условия, когда это неравенство может быть нарушено. Последующие биты кода могут увеличить значение s на величину $\alpha < 1$, которая, будучи умножена на δ , даст величину $\alpha\delta < \delta$. Если I_c лишь на единицу меньше $\widehat{\vartheta}(u)$, а $R_c > d - \delta$, то учет поправки $\alpha\delta$ может привести к прибавлению единицы к I_c , что нарушит рассматриваемое неравенство в пользу неравенства $\vartheta(w) \leq I_c < \widehat{\vartheta}(w)$, где w есть следующий за u символ алфавита A , $\vartheta(w) = \widehat{\vartheta}(u)$.

Ситуация, когда точка s внутри интервала $[l, h)$ приводит к такой неопределенности выбора между символами u и w , возможна только вследствие того, что при кодировании был выбран интервал V^+ для u либо V^- для w . Для разрешения неопределенности мы должны масштабировать единичный интервал $[c, c+1)$ (фактически взять следующие t бит кодовой последовательности) и выполнить декодирование для двухбуквенного алфавита $A' = \{u, w\}$ с распределением вероятностей

$$\vartheta'(u) = 0, \quad \widehat{\vartheta}'(u) = \vartheta'(w) = R_{uw}, \quad \widehat{\vartheta}'(w) = \delta,$$

где

$$R_{uw} = d\widehat{\vartheta}(u) \pmod{\delta} = d\vartheta(w) \pmod{\delta}.$$

Можно избежать излишних вычислений, если заметить, что

$$R_{uw} = d - R_c.$$

Действительно,

$$\begin{aligned} \widehat{\vartheta}(u) &= I_c + 1 = \frac{(c-l)\delta}{d} - \frac{R_c}{d} + 1, \\ R_{uw} &= (c-l)\delta - R_c + d \pmod{\delta} \\ &= ((c-l)\delta \pmod{\delta} + (d - R_c) \pmod{\delta}) \pmod{\delta} \\ &= 0 + (d - R_c) \pmod{\delta} = d - R_c. \end{aligned}$$

Основная идея метода декодирования состоит в том, что, узнав первый закодированный символ, декодер может воспроизвести действия кодера и таким образом подготовить основу для декодирования следующего символа. Поэтому дадим описание метода декодирования с сохранением максимальной симметрии по отношению к методу кодирования.

Алгоритм декодирования интервала

Масштабирование

Вход: интервал $[l, h)$

и текущие t бит кодовой последовательности в c .

$g := h - 1;$

пока $l_1 = g_1$

$c := 2c +$ очередной бит кодовой последовательности,

$l := 2l, \quad g := 2g + 1 \pmod{2^t};$

пока $l_2 = 1$ и $g_2 = 0$

$c := 2(c - 2^{t-1}) +$ очередной бит кодовой последовательности,

$l := 2l, \quad g := 2g + 1 \pmod{2^t};$

$l_1 := 0, \quad g_1 := 1;$

$h := g + 1.$

Инициализация

$c :=$ первые t бит кодовой последовательности (только первый раз)

$l := 0, \quad h := 2^t.$

Декодирование очередного символа

(параметры: распределение ϑ на алфавите A , δ)

$d := h - l,$

$I_c = \lfloor (c - l)\delta/d \rfloor, \quad R_c = (c - l)\delta \pmod{d};$

найти символ u такой, что $\vartheta(u) \leq I_c < \hat{\vartheta}(u);$

если $(I_c = \hat{\vartheta}(u) - 1)$ и $(d - R_c < \delta)$, то

$l := c, \quad h := c + 1,$ масштабировать $[l, h);$

$w :=$ следующая за u буква в алфавите A ($\hat{\vartheta}(u) = \vartheta(w)$);

выполнить декодирование, используя вместо A и ϑ

алфавит $A' = \{u, w\}$ с распределением вероятностей

$\vartheta'(u) = 0, \quad \hat{\vartheta}'(u) = \vartheta'(w) = d - R_c, \quad \hat{\vartheta}'(w) = \delta;$

иначе

u есть закодированный символ;

$I_l := \lfloor d\vartheta(u)/\delta \rfloor, \quad R_l := d\vartheta(u) \pmod{\delta};$

$I_h := \lfloor d\hat{\vartheta}(u)/\delta \rfloor, \quad R_h := d\hat{\vartheta}(u) \pmod{\delta};$

$h := l + I_h,$

$l := l + I_l$, если $R_l \neq 0$, то $l := l + 1,$

масштабировать $[l, h);$

конец декодирования очередного символа.

*Удаление кодовых бит, полученных в результате
омофонного кодирования заключительного интервала*

Вход: нормализованный интервал $[l, h)$.

Если не выполняется условие ($l = 0$ и $h = 2^t$), то
удалить из s те биты, которые определяют
попадание s в интервал $[l, h)$
(алгоритм приведен в приложении).

Остановимся теперь на оценках основных свойств предложенного метода.

Лемма 1. Длина d нормализованного интервала $[l, h)$ удовлетворяет неравенству

$$d \geq 2^{t-2} + 2. \quad (4)$$

Доказательство. От $[l, h)$ перейдем к замкнутому интервалу $[l, g]$, $g = h - 1 + 0,111\dots$. Тогда

$$d = h - l = \lfloor g \rfloor + 1 - l.$$

Рассматривая l и g как двоичные числа, получаем три варианта нормализованных интервалов: $[00*, 11*]$, $[00*, 10*]$ и $[01*, 11*]$, где $*$ обозначает $t - 2$ произвольные двоичные цифры. Очевидно, что длина интервала в первом случае больше, чем во втором, а второй и третий случаи эквивалентны. Для второго случая имеем

$$l \leq 2^{t-2} - 1, \quad \lfloor g \rfloor \geq 2 \cdot 2^{t-2}.$$

Отсюда

$$d \geq 2 \cdot 2^{t-2} + 1 - (2^{t-2} - 1) = 2^{t-2} + 2.$$

Лемма 1 доказана.

Теорема 1. Пусть имеется сообщение $U = u_1 u_2 \dots u_L$, порожденное некоторым источником из букв алфавита $A = \{a_1, a_2, \dots, a_N\}$, и пусть для каждого символа $u \in U$ известно распределение вероятностей $P(a_1)$, $P(a_2), \dots, P(a_N)$. Если кодировать сообщение U описанным методом арифметического кодирования с разделением интервала, причем для представления интервала использовать t -битные целые числа, $t \geq 4$, то справедливы следующие утверждения:

1. Избыточность кода r на символ сообщения u удовлетворяет неравенству

$$r(u) < 8Nt2^{-t} + 3/L.$$

2. Объем M памяти кодера (декодера) и среднее время кодирования (декодирования) T при $t \rightarrow \infty$ удовлетворяют соотношениям

$$M = O(t),$$

$$T = O(t \log t \log \log t).$$

Доказательство. Обозначим через $C(u)$ длину кодового слова для символа u , через $E[C(u)]$ ожидаемую (среднюю) длину кодового слова, через $H(u)$ энтропию на символ источника.

При арифметическом кодировании, основанном на точном вычислении интервала, избыточность возникает только из-за ограничения размера L кодируемого блока символов и, как было показано в [3, 11], при омофонном кодировании заключительного интервала не превосходит 3 бит на блок. При неограниченном возрастании размера блока избыточность стремится к нулю. Это позволяет отделить ту часть избыточности, которая связана с ограничением размера блока, от той ее части, которая связана с вычислением интервала, и записать

$$r(u) = E[C(u)] - H(u) + 3/L.$$

(В этом случае $C(u)$ можно считать длиной кодового слова при условии отсутствия ограничения длины блока.) Введем дополнительные обозначения:

$$E[C(u)] = \sum_{i=1}^N P(a_i)C(a_i) = \sum_{i=1}^N e(a_i).$$

Для арифметического кодирования с точным вычислением интервала имеем

$$e(u) = -P(u) \log P(u).$$

Для метода с разделением интервала каждый символ u фактически представляется совокупностью омофонов $v, v_1^-, v_1^+, v_2^-, v_2^+, \dots$ с вероятностями $P(v), P(v_1^-), P(v_1^+), \dots$, сумма которых равна $P(u)$, причем для каждого омофона интервал вычисляется без потери точности. Это значит, что

$$e(u) = -[P(v) \log P(v) + P(v_1^-) \log P(v_1^-) + P(v_1^+) \log P(v_1^+) + \dots]. \quad (5)$$

Так как в схеме кодирования омофоны v_1^- и v_1^+ появляются вследствие разделения нормализованного интервала, а длина последнего удовлетворяет неравенству (4), то вероятности $P(v_1^-)$ и $P(v_1^+)$ не могут быть больше $2^{-(t-2)}$. Последующие омофоны v_2^- и v_2^+ появляются уже при разделении полного интервала $[0, 2^t)$. Поэтому их вероятности не превышают $2^{-(t-2)} \cdot 2^{-t}$ и т. д. Подставляя в (5) вместо вероятностей омофонов их верхние оценки, получаем

$$\begin{aligned} e(u) &< -P(v) \log P(v) + 2 \cdot 2^{-(t-2)}(t-2) \\ &\quad + 2 \cdot 2^{-(2t-2)}(2t-2) + 2 \cdot 2^{-(3t-2)}(3t-2) + \dots \\ &= -P(v) \log P(v) + 8t[2^{-t} + 2 \cdot 2^{-2t} + 3 \cdot 2^{-3t} + \dots] - 16[2^{-t} + 2^{-2t} + 2^{-3t} + \dots] \\ &= -P(v) \log P(v) + 8t \frac{2^{-t}}{(1-2^{-t})^2} - 16 \frac{2^{-t}}{1-2^{-t}}. \end{aligned}$$

Принимая во внимание неравенства

$$P(v) \leq P(u), \quad P(u) < 2^{-(t-2)} + P(v) + 2^{-(t-2)}, \quad t \geq 4,$$

исследуем разность $P(u) \log P(u) - P(v) \log P(v)$. Эта разность возрастает с ростом $P(u)$. Поэтому можно записать

$$\begin{aligned} P(u) \log P(u) - P(v) \log P(v) &< \lim_{P(u) \rightarrow 1} (P(u) \log P(u) - P(v) \log P(v)) \\ &< - \lim_{P(u) \rightarrow 1} ((P(u) - 8 \cdot 2^{-t}) \log(P(u) - 8 \cdot 2^{-t})) \\ &= -(1 - 8 \cdot 2^{-t}) \log(1 - 8 \cdot 2^{-t}) \leq \frac{1}{\ln 2} (1 - 8 \cdot 2^{-t}) 8 \cdot 2^{-t}. \end{aligned}$$

Поэтому

$$\begin{aligned} r(u) &= \sum_{i=1}^N e(a_i) - \sum_{i=1}^N -P(a_i) \log P(a_i) + 3/L \\ &< \sum_{i=1}^N \left(P(a_i) \log P(a_i) - P(v) \log P(v) + 8t \frac{2^{-t}}{(1 - 2^{-t})^2} - 16 \frac{2^{-t}}{1 - 2^{-t}} \right) + 3/L \\ &< \sum_{i=1}^N 8t 2^{-t} \left[\frac{1}{t \ln 2} + \frac{1}{(1 - 2^{-t})^2} - \frac{2}{t(1 - 2^{-t})} \right] + 3/L \\ &< \sum_{i=1}^N 8t 2^{-t} + 3/L = 8Nt 2^{-t} + 3/L. \quad (6) \end{aligned}$$

То, что выражение в квадратных скобках в (6) меньше единицы, доказывается при замене $1/\ln 2$ его верхней оценкой $3/2$ и в предположении, что $t \geq 4$.

Из описания метода видно, что для его работы требуется t -битная арифметика целых положительных чисел, что подразумевает выполнение операций типа сложения и сдвига с t - и $2t$ -битными операндами, умножения двух t -битных чисел с образованием $2t$ -битного результата и деления $2t$ -битного числа на t -битное с получением t -битных целой части и остатка.

В целях упрощения описания мы допустили, что некоторые из переменных могут принимать значение 2^t , что, вообще говоря, требует $t + 1$ двоичного разряда. В реальных программах это значение во всех случаях может быть закодировано нулем при достаточно простой модификации вычислений с участием таких переменных.

Соотношение $M = O(t)$ достаточно очевидно.

Теперь оценим среднее время кодирования. На каждой итерации процесса кодирования производится вычисление границ интервала, выбор подынтервала и масштабирование. Поскольку вероятность каждой

следующей итерации экспоненциально снижается, среднее число итераций ограничено константой. Для вычисления границ интервала требуется две t -битных операции умножения и деления, сложность которых (при использовании быстрых методов) не превосходит $O(t \log t \log \log t)$ (см. метод Шёнхаге — Штрассена в [1]). Метод вероятностного выбора подынтервала, описываемый в разд. 3, требует в среднем не более трех итераций с несколькими простыми операциями в каждой. При масштабировании каждая итерация любого из его циклов приводит к формированию одного кодового бита. Поэтому среднее число итераций масштабирования при кодировании символа u будет равно $H(u) + r(u)$. Очевидно, что доминирующим по трудоемкости оказывается этап вычисления границ интервала.

Аналогичные рассуждения справедливы и для декодирования. Теорема 1 доказана.

Следствие 1. Объем памяти M и время работы T как функции избыточности r при $r \rightarrow 0$ удовлетворяют соотношениям

$$M = O(\log(1/r)),$$

$$T = O(\log(1/r) \log \log(1/r) \log \log \log(1/r)).$$

Доказательство. Положим разрядность счетчика S равной t и возьмем длину блока $L = 2^t/t$, что удовлетворяет ограничению (3). Тогда $r < 8Nt2^{-t} + 3t2^{-t}$, что позволяет записать

$$r = O(t2^{-t}), \quad t \rightarrow \infty.$$

Для любого t , большего некоторого t^* , можно найти константу $c > 1$ такую, что выполнится неравенство $t2^{-t} < 2^{-t/c}$. Следовательно, $r = O(2^{-t/c})$. Поэтому

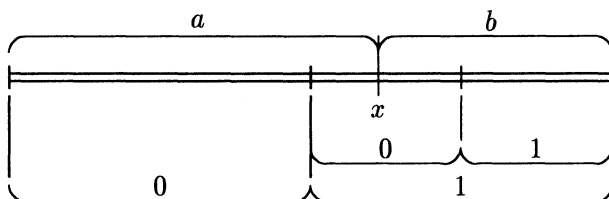
$$\log(1/r) = O(t/c) = O(t),$$

что с учетом второй части теоремы 1 приводит к утверждению следствия 1.

3. Реализация случайного выбора интервала

В описанном методе рандомизации требуется производить случайный выбор одного из трех или одного из двух интервалов с вероятностью, пропорциональной его длине. Мы предлагаем простой способ решения этой задачи. Начнем со случая двух интервалов.

Пусть требуется выбрать один из двух отрезков A, B с длинами a, b . Введем отрезок Z , составленный из отрезков A, B . Его длина $z = a + b$ (рис. 6).


 Рис. 6. Вероятностный выбор одного из отрезков A, B

Разделим отрезок Z пополам и выделим его левую (или правую) половину в зависимости от значения полученного случайного бита. Если выделенная половина целиком принадлежит одному из отрезков A, B , то этот отрезок будем считать выбранным. В противном случае сократим отрезок Z до его выделенной половины и повторим весь процесс сначала.

Очевидно, что вероятности выбора A или B будут пропорциональны их длинам. Повышения точности вычислений с каждым последующим делением Z можно избежать, если каждый раз масштабировать выделенную половину Z до полной длины Z . Получаем следующий алгоритм:

$z := a + b, \quad x := a;$

повторять следующие действия, пока не будет сделан выбор:

$\beta :=$ случайный бит, $x := 2x;$

если $\beta = 0$ и $x \geq z$, то выбран отрезок A ;

если $\beta = 1$ и $x \leq z$, то выбран отрезок B ;

если $x > z$, то $x := x - z$.

Оценим среднее число используемых случайных бит $N_r^{(2)}$. Заметим, что длина одного из отрезков обязательно больше либо равна половине Z . Поэтому с вероятностью $1/2$ выбор будет сделан с помощью одного случайного бита. Если на первом шаге выбор не был сделан, то предыдущее рассуждение может быть повторено и для второго шага и т. д. В итоге получаем

$$N_r^{(2)} \leq 1 \times \frac{1}{2} + 2 \times \frac{1}{4} + 3 \times \frac{1}{8} + \dots = 2.$$

Теперь рассмотрим задачу выбора одного из трех отрезков A, B, C с длинами a, b, c . Будем считать, что отрезки упорядочены по длине следующим образом: $a \geq c \geq b$. Составим отрезок $Z = A \cdot B \cdot C$, $z = a + b + c$. Для реализации выбора будем использовать тот же процесс деления Z .

Если $a < b + c$, то середина Z попадает в B и после первого деления задача будет сведена к случаю двух отрезков. Число случайных бит

$$N_r^{(3)} = 1 + 2 = 3.$$

Если $a \geq b + c$, то можно показать, что среднее число случайных бит будет меньше 3. Например, при выборе омофонов символа b (см. рис. 2) имеем $a = 4, b = 1, c = 1; z = 6$. После первого деления мы либо

выбираем A , либо получаем вновь три отрезка с соотношением длин 1:1:1 и для дальнейшего выбора требуется 3 бита. В этом случае $N_r = 1 \times \frac{1}{2} + (1+3) \times \frac{1}{2} = 2\frac{1}{2}$.

Для получения более точной оценки заметим, что для любых a , b и c существует неотрицательное целое число k такое, что справедливы неравенства

$$(2^k - 1)(b + c) \leq a < (2^{k+1} - 1)(b + c). \quad (7)$$

Легко показать, что в этом случае

$$N_r^{(3)} \leq 1 \times \frac{1}{2} + 2 \times \frac{1}{4} + \dots + k \times \frac{1}{2^k} + (k+3) \times \frac{1}{2^k} = 2 + \frac{1}{2^k}$$

и из правой части неравенства (7) следует

$$2^k > \frac{z}{2(b+c)}. \quad (8)$$

При кодировании символа u имеем

$$z = d\rho(u), \quad b + c < 2\delta, \quad d > 2^{t-2}, \quad (9)$$

где d — длина нормализованного интервала, а $\rho(u)/\delta = P(u)$ есть вероятность появления символа u (см. алгоритм кодирования).

Объединяя (8) и (9), получаем

$$2^k > \frac{d\rho(u)}{4\delta} > \frac{2^t}{16} P(u),$$

откуда следует

$$N_r^{(3)}(u) < 2 + \frac{16 \cdot 2^{-t}}{P(u)}.$$

Усредняя оценку количества случайных бит по множеству букв алфавита источника, получаем

$$E[N_r^{(3)}(u)] = \sum_{i=1}^N P(a_i) N_r^{(3)}(a_i) < \sum_{i=1}^N (2P(a_i) + 16 \cdot 2^{-t}) = 2 + 16N2^{-t}.$$

В описанном методе рандомизации необходимость выбора интервала при кодировании одного символа может возникать многократно. Причем при первом разделении буквы алфавита источника распределены на нормализованном интервале $[l, h)$ и выбор делается (в общем случае) между тремя интервалами. При последующих разделениях выбор всегда производится между двумя интервалами, распределенными на полном интервале $[0, 2^t)$. Необходимо также учесть омофонное кодирование интервала при завершении кодовой последовательности, для проведения которого в соответствии с [3, 11] требуется в среднем не более 12 случайных бит. Таким образом, получаем

$$\begin{aligned} N_r &< E[N_r^{(3)}(u)] + (N-1)2^{-(t-2)}(2 + 2^{-t}(2 + \dots)) + 12/L \\ &= E[N_r^{(3)}(u)] + 8(N-1)[2^{-t} + 2^{-2t} + 2^{-3t} + \dots] + 12/L \\ &= 2 + \frac{16N}{2^t} + \frac{8(N-1)}{2^t - 1} + 12/L < 2 + 24N2^{-t} + 12/L. \end{aligned}$$

Мы доказали следующее утверждение.

Теорема 2. Для метода рандомизации с разделением интервала существует метод случайного выбора, для которого среднее число случайных бит N_r на один символ источника u удовлетворяет неравенству

$$N_r(u) < 2 + 24N2^{-t} + 12/L,$$

где N — число букв в алфавите источника, t — точность представления интервала и L — длина блока символов сообщения.

С учетом первого утверждения теоремы 1 становится очевидным

Следствие 2. Среднее число случайных бит и избыточность r при $r \rightarrow 0$ удовлетворяют неравенству

$$N_r(u) < 2 + r(u).$$

Приложение

Алгоритм омофонного кодирования интервала $[l, h)$

Вход: нормализованный интервал $[l, h)$.

Выход: последовательность случайных бит $\beta_0\beta_1\dots\beta_k$, определяющая код выбранного омофона

(все вычисления выполняются по модулю 2^t).

$l' := l, \quad h' := h, \quad \beta_0 := \text{случайный бит}, \quad k := 0;$

если $\beta_0 = 0$, то

пока $l' \neq 0$ и $\beta_k = l'_1$

$l' := 2l',$

если $l' \neq 0$, то $k := k + 1, \beta_k := \text{случайный бит};$

если $l' = 0$ или $\beta_k > l'_1$, то конец (последовательность β сформирована),

в противном случае повторить все сначала;

иначе ($\beta_0 = 1$)

пока $h' \neq 0$ и $\beta_k = h'_1$

$h' := 2h',$

если $h' \neq 0$, то $k := k + 1, \beta_k := \text{случайный бит};$

если $\beta_k < h'_1$, то конец (последовательность β сформирована),

в противном случае повторить все сначала.

Алгоритм исключения из s бит, определяющих попадание s в интервал $[l, h)$

Вход: нормализованный интервал $[l, h)$.

Выход: биты новой кодовой последовательности в s

(все вычисления выполняются по модулю 2^t).

если $s_1 = 0$, то $x := l;$

если $s_1 = 1$, то $x := h;$

пока $x \neq 0$ и $s_1 = x_1$

$x := 2x,$

если $x \neq 0$, то $c := 2c +$ очередной бит кодовой последовательности;

$c := 2c +$ очередной бит кодовой последовательности.

ЛИТЕРАТУРА

1. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. М.: Мир, 1979.
2. Рябко Б. Я. Эффективный метод кодирования источников информации, использующий алгоритм быстрого умножения // Проблемы передачи информации. 1995. Т. 31, вып. 1. С. 3–12.
3. Рябко Б. Я., Фионов А. Н. Быстрый метод полной рандомизации сообщений // Проблемы передачи информации. 1997. Т. 33, вып. 3.
4. Шеннон К. Работы по теории информации и кибернетике. М.: Изд-во иностр. лит., 1963. С. 333–402.
5. Bell T. C., Cleary J. G., Witten I. H. Text compression. Englewood Cliffs, NJ: Prentice Hall, Inc., 1990.
6. Günther Ch. G. A universal algorithm for homophonic coding // Advances in Cryptology — EUROCRYPT'88. Berlin: Springer-Verl., 1988. P. 405–414. (Lecture Notes in Comput. Sci.; V. 330).
7. Jendal H. N., Kuhn Y. J. B., Massey J. L. An information-theoretic treatment of homophonic substitution // Advances in Cryptology — EUROCRYPT'89. Berlin: Springer-Verl., 1990. P. 382–394. (Lecture Notes in Comput. Sci.; V. 434).
8. Massey J. L. An introduction to contemporary cryptology // Proc. of the IEEE, 1988. V. 76. P. 533–549.
9. Rissanen J., Langdon G. G. Universal modeling and coding // IEEE Trans. Inform. Theory. 1981. V. 27, N 1. P. 12–23.
10. Ryabko B. Y. Fast and effective coding of information sources // IEEE Trans. Inform. Theory. 1994. V. 40, N 1. P. 96–99.
11. Ryabko B. Y., Fionov A. N. A fast and efficient homophonic coding algorithm // Algorithms and Computation. Berlin: Springer, 1996. P. 427–438. (Lecture Notes in Comput. Sci.; V. 1178).

Адрес автора:

Сибирская государственная
академия телекоммуникаций
и информатики,
ул. Кирова, 86,
630125 Новосибирск, Россия.
E-mail: fionov@neic.nsk.su

Статья поступила

17 декабря 1996 г.