# Efficient Homophonic Coding

Boris Ryabko and Andrei Fionov

***Abstract*—Homophonic coding, or homophonic substitution, is referred to as a technique that contributes to reliability of the secret-key cipher systems. Its main goal is to convert the plaintext into a sequence of completely random (equiprobable and independent) code letters. In solving this problem three characteristics are to be considered: i) redundancy, defined as the difference between the mean codeword length and the source entropy, ii) an average number of random bits used in encoding, and iii) complexity of the encoder and decoder, measured by memory size (in bits) and computation time (in bit operations). A class of homophonic codes is suggested for which both the redundancy and the average number of random bits can be made as small as required with nonexponential growth of memory size and roughly logarithmic growth of computation time.**

***Index Terms*— Computational complexity, homophonic coding, randomization, secret key cryptosystems, source coding.**

## I. Introduction

Homophonic coding, or homophonic substitution, is known to be a kind of message randomization that aims to convert a source message, consisting of nonuniformly distributed letters of some alphabet, into a uniquely decodable sequence of "more uniformly" distributed code symbols. The uniformity of distribution means that all the symbols are equiprobable and independent, or, in other words, completely random. Randomization is said to be perfect if it achieves an exactly uniform distribution, i.e., produces a completely random output. Only those homophonic coding schemes that ensure perfect randomization will be dealt with in the present correspondence.

The main application field of message randomization is cryptography. The grounds for this were established by Shannon in his pioneering work on secrecy systems [1]. Shannon introduced a notion of the key-equivocation function $f(n)$, which was defined to be the conditional entropy of the secret key given the first $n$ digits of ciphertext, i.e., $f(n) = H(Z|Y^n)$, where $Z$ denotes the secret key and $Y^n$ denotes the first $n$ digits of the ciphertext. He called a secret-key cipher system strongly ideal if $f(n)$ is constant, i.e., $H(Z|Y^n) = H(Z)$ for all $n$, which is equivalent to the statement that the ciphertext is statistically independent of the secret key. The strongly ideal cryptosystem ensures that the secret key cannot be reconstructed by a cryptoanalyst using a ciphertext-only attack, in spite of whatever long sequence of ciphertext he may have. Shannon also noticed that if there existed an "artificial" language whose letters were equiprobable and independent, then a simple cipher would produce a strongly ideal cryptosystem (regardless of the statistics for the secret key). Message randomization is intended to simulate the artificial language mentioned by Shannon. The benefits of this simulation for cryptography are studied in [2] and [3].

To clarify the main idea of conventional homophonic coding consider an example. Let a Bernoulli source generate letters over the alphabet $A = \{a, b\}$ with probabilities $P(a) = 3/4$, $P(b) = 1/4$. Encode source symbols according to the following mapping:

$$a \rightarrow \begin{cases} v_1 \rightarrow 00 & \text{with probability } 1/3 \\ v_2 \rightarrow 01 & \text{with probability } 1/3 \\ v_3 \rightarrow 10 & \text{with probability } 1/3 \end{cases}$$
$$b \rightarrow \quad v_4 \rightarrow 11 \quad \text{with probability } 1.$$

Here $v_1, v_2, v_3, v_4$ are homophones that substitute the source symbols. Each homophone is encoded by a 2-bit codeword. We obtain a code sequence indistinguishable from a random one since the probability of any 2-bit combination is equal to $1/4$. This coding scheme, however, has a disadvantage of, in general, not minimizing the average codeword length and being applicable only to sources whose symbol probabilities are the ratio between a positive integer and a power of 2.

In [2], a variable-length homophonic coding scheme, which does not have that disadvantage, was proposed. It claims that the mapping for the same source be done as follows:

$$a \rightarrow \begin{cases} v_1 \rightarrow 0 & \text{with probability } 2/3 \\ v_2 \rightarrow 10 & \text{with probability } 1/3 \end{cases}$$
$$b \rightarrow \quad v_3 \rightarrow 11 \quad \text{with probability } 1. \tag{1}$$

This scheme tackles the problem of perfect randomization, as well, the mean codeword length being reduced. Besides, the method may be applied to sources with arbitrary rational symbol probabilities.

In [3], an optimum homophonic coding scheme was defined as a scheme minimizing the mean codeword length for a given source. The increase of the message length produced by a randomization method may be measured by redundancy, denoted henceforth by $r$, which is defined as the difference between the mean codeword length and the source entropy. To make a choice among homophones, encoding methods use independent random bits obtained from a table or a generator. It has been shown in [3] that if the homophones have distinct choice probabilities for every source symbol (as in the above example) then the homophonic coding scheme is optimal, the redundancy does not exceed 2 bits and not more than four random bits on average are required to pick a homophone. A more accurate proof of this result has been given in [4]. Note that the redundancy of optimum homophonic coding is only one bit greater than that for Huffman coding. Thus one additional bit of redundancy is the payment for the randomness of code sequence.

One often may face applications where redundancy and, perhaps, a number of random bits used in encoding, should be decreased. The problem of decreasing redundancy is common for information theory and aims to lessen the length of encoded message. The problem of decreasing the number of random bits is not so widely known. There exist two opposite points of view on the problem of random bit generation. From one point of view, it is quite a simple task to generate pseudo-random numbers and we have a lot of computer programs to do it fast. From another point of view, generating truly random numbers is a hard task. More exactly, a computer program, whose length is shorter than that of a random bit sequence it aims to generate (which is based on the concept of Kolmogorov complexity), does not exist. Of course, there are many who adhere to an intermediate point of view on the problem. Therefore, it would be useful to have a class of methods which make it possible to use as few random bits as one may desire. Of course, one may expect that the lower the number of random bits to be consumed the greater are complexities of the encoder and decoder.

To decrease redundancy one may encode blocks of symbols. However, conventional methods of block coding lead to exponential

Manuscript received November 9, 1997; revised October 15, 1998. This work was supported by the Russian Foundation of Basis Research under Grant 99-01-00586.

The authors are with SibSATIS, Novosibirsk, 630102, Russia.

Communicated by D. Stinson, Associate Editor for Complexity and Cryptography.

Publisher Item Identifier S 0018-9448(99)05864-2.

growth of the memory size (or computation time) as the length of a block increases, and hence are intractable. We suggest a homophonic coding method, based on efficient block coding, which requires only linear growth of memory size when increasing the length of a block (preliminary versions of the method were presented in [5] and [6]). The method allows to obtain arbitrarily low redundancy $r$ per source symbol with memory size and computation time growing as

$$O(1/r) \text{ and } O(\log^2 1/r \log\log 1/r) \tag{2}$$

respectively, as $r \to 0$. In the process of encoding not more than $4r$ random bits per symbol are used on average, which corresponds to the case when random bits are hard to generate.

When random bits are easily obtained, we can suggest a still more efficient method of homophonic coding based on arithmetic coding (for arithmetic coding details see, e.g., [7] and [8]; see also the paper [9] whose author was apparently the first to apply the technique to the homophonic coding problem). Our method, when providing arbitrarily small redundancy, requires the memory size and computation time to grow as

$$O(\log 1/r) \text{ and } O(\log 1/r \log\log 1/r \log\log\log 1/r) \tag{3}$$

respectively, as $r \to 0$. We need, however, about $2 + r$ random bits for encoding each source symbol.

The two mentioned approaches are combined in a class of homophonic coding methods that allows to construct homophonic codes under simultaneously imposed restrictions both on the redundancy and the (average) number of random bits to be used. This solution corresponds to the centrist position on the problem of random bit generation and its computational complexity lies between (2) and (3).

For the sake of simplicity, we confine ourselves only to Bernoulli sources and a binary encoding alphabet. However, our methods can easily be applied to Markov sources by using the standard techniques developed in source coding theory. If a Markov source has the connectivity $\nu$, i.e., the probability of every source symbol depends on $\nu$ preceding symbols, and the size of the source alphabet is $|A|$ then we need $O(|A|^{\nu+1})$ bits of memory to store conditional probabilities. Nevertheless, the asymptotic estimates of the memory size and computation time given by (2) and (3) still remain correct.

The correspondence is organized as follows. In the next section, we give necessary notations and present, by means of simple examples, the main ideas of our approach. In Section III, we describe a letterwise homophonic coding method on which the further constructions are based. In Sections IV and V, we describe two approaches corresponding to two opposite points of view on the random bit generation problem. Finally, in Section VI, we give a synthesis of the two approaches corresponding to a general case when both the redundancy and the number of random bits are restricted.

## II. Notations, Definitions, and Basic Ideas

Let there be given a Bernoulli source generating letters over the alphabet $A = \{a_1, a_2, \cdots, a_N\}$ with probabilities $P(a_1)$, $P(a_2)$, $\cdots$, $P(a_N)$, represented by rational numbers

$$P(a_1) = \frac{\rho(a_1)}{\delta} \qquad P(a_2) = \frac{\rho(a_2)}{\delta}, \cdots, P(a_N) = \frac{\rho(a_N)}{\delta}.$$

Define cumulative probabilities $Q(a_1), Q(a_2), \cdots, Q(a_N)$ as follows:

$$Q(a_1) = 0, \quad Q(a_i) = \sum_{j<i} P(a_j), \qquad i = 2, 3, \cdots, N. \tag{4}$$

Introduce an auxilliary value $\hat{Q}(u) = Q(u) + P(u)$. Consider $Q(u)$ and $\hat{Q}(u)$ as rational numbers: $Q(u) = \vartheta(u)/\delta$, $\hat{Q}(u) = \hat{\vartheta}(u)/\delta$. Let $\rho, \vartheta, \hat{\vartheta}$, and $\delta$ be $\tau$-bit nonnegative integers (for ease of designation, allow $\hat{\vartheta}$ and $\delta$ to take the value $2^\tau$). Our aim is to encode a message $U = u_1 u_2 \cdots u_L$ generated by the source. The message $U$ may be treated as a letter generated over the alphabet $A^L$ with probability $P(U) = P(u_1)P(u_2)\cdots P(u_L)$. Set a lexicographic order over $A^L$ and define cumulative probabilities

$$Q(a_1 a_1 \cdots a_1) = 0, \quad Q(U) = \sum_{V<U,\, V \in A^L} P(V).$$

Let again $\hat{Q}(U) = Q(U) + P(U)$.

The keynote of our approach to efficient homophonic coding is describing the encoding scheme in terms of intervals. Consider the half-open interval $[0, 1)$ which we shall call an entire range. Cumulative probability distribution implies a mapping of each source letter into a correspondent interval:

$$a_1 \to [Q(a_1), \hat{Q}(a_1)), \qquad a_2 \to [Q(a_2), \hat{Q}(a_2)), \cdots, a_N \to [Q(a_N), \hat{Q}(a_N)).$$

All these intervals are distinct and cover the entire range because $Q(a_1) = 0$, $\hat{Q}(a_i) = Q(a_{i+1})$ for all $i < N$, $\hat{Q}(a_N) = 1$, which can be easily deduced from (4). To ensure perfect randomization we must partition each interval into a number of homophone intervals so that i) the size $\zeta$ of each homophone interval be a negative power of 2 and ii) each interval of size $\zeta$ could be encoded exactly in $\log \zeta$ bits (here and below $\log x = \log_2 x$). To make such a partition, the following method can be used (forget for the time being that a number of homophones may be infinite). Divide the entire range into two parts ascribing 0 to the left part and 1 to the right part. For both parts do the following. If a part fits entirely within the interval allocated to some letter then this part is a homophone interval for that letter. Otherwise, continue the process of dividing for a part just as for the entire range. Eventually we obtain a partition of all letter intervals into homophone intervals. The ascribed bits form a codeword for each homophone. Each homophone interval is to be picked with probability proportional to its size.

To exemplify this method consider the encoding scheme (1). Cumulative probabilities of the letters dictate the following initial mapping: $a \to [0, 3/4)$, $b \to [3/4, 1)$. Dividing the entire range gives the intervals $[0, 1/2)$ and $[1/2, 1)$ with ascribed bits 0 and 1, respectively. The interval $[0, 1/2)$ fits entirely within the interval $[0, 3/4)$ allocated to the letter $a$, hence it is a homophone interval for $a$. Dividing the interval $[1/2, 1)$ gives $[1/2, 3/4)$ and $[3/4, 1)$, the ascribed bits being 10 and 11, respectively. Both intervals are homophonic, the first one corresponds to the letter $a$, and the second one corresponds to the letter $b$. As a result, we have the following mapping:

$$a \to [0, 3/4) \to \begin{cases} [0, 1/2) & \to 0 \quad \text{with probability } 2/3 \\ [1/2, 3/4) & \to 10 \quad \text{with probability } 1/3 \end{cases}$$
$$b \to [3/4, 1) \to \quad [3/4, 1) \quad \to 11 \quad \text{with probability } 1.$$

This is essentially the same as (1). It is important that partitioning for every letter may be obtained independently. This allows not to construct the whole encoding table. Partitioning may be combined with homophone selection, which solves the problem of infinite number of homophones. A detailed description of the method is given in Section III. It is worth noting here that the described partitioning cannot guarantee distinct probabilities for all homophones and hence the scheme is not optimal from the viewpoint of minimizing average codeword length. We prove, however, that the maximum mean redundancy is only one bit greater than that for optimum encoding. We show also that (in the worst case) a maximum of 12 random bits on average are needed for homophone selection instead of four bits in case of an optimum scheme.

In arithmetic coding (see, e.g., [7] and [8]), as well as in fast block coding from [10], a message $U$ is represented by an interval $[Q(U),\, \hat{Q}(U))$ allocated to it in the cumulative probability distribution. A codeword is constructed as a number within the interval. Every source message has an interval allocated to it within the initial range. If the interval bounds are computed precisely then the interval size is proportional to the message probability. A set of such intervals covers the entire initial range. For the purpose of randomization we may perform homophonic substitution for the interval of a message just like we do for a single letter. It is obvious that a code sequence produced in this way is completely random.

The idea of precise computation of the interval for a message leads us to a method which we shall call block homophonic coding (BHC for brevity). Let, for example, there be given the message $U = aba$ generated by the source defined in Section I. To encode the message, compute

$$\begin{aligned} &Q(aba) = P(aaa) + P(aab) = 27/64 + 9/64 = 36/64\\ &P(aba) = 9/64\\ &\hat{Q}(aba) = 45/64. \end{aligned}$$

Partitioning of the interval $[36/64,\, 45/64)$ gives three homophone intervals $[36/64,\, 40/64)$, $[40/64,\, 44/64)$, and $[44/64,\, 45/64)$ with ascribed bits 1001, 1010, and 101100, respectively. As a result we have the following mapping:

$$aba \to \begin{cases} 1001 & \text{with probability } 4/9\\ 1010 & \text{with probability } 4/9\\ 101100 & \text{with probability } 1/9. \end{cases}$$

The main part of the BHC algorithm is fast computation of the interval which has to be done with the arithmetic precision being increased from $\tau$ to $L\tau$ bits, where $L$ is the length of a message (or block). Although the method copes well with the problem, one would like to search for a more efficient method to keep the arithmetic precision.

We suggest a method based on incremental arithmetic coding. The interval for a message is obtained by narrowing some initial interval by each successive symbol of the message. To retain constant arithmetic precision we suggest a technique called "interval splitting." It assumes that the interval is represented by homophonic intervals and only one of them is picked for further narrowing. This is accomplished by scaling, a technique well known in practical arithmetic coding (see, e.g., [8]), which is an operation intended to exclude from the numbers representing the interval bounds those digits that cannot be affected by further computations. Taken together, splitting and scaling guarantee both perfect statistical properties of the code and constant arithmetic precision. The method will be referred to as the arithmetic coding with interval splitting (for brevity, ACIS) method. To emphasize that the arithmetic precision is constant, we transfer the interval from real to integer numbers (the entire range $[0,\, 1)$ is scaled to $[0,\, 2^t)$, $t \geq \tau + 2$ for reasons discussed in Section V) and perform splitting every time the interval bound values become noninteger. To let the peculiarities of the ACIS encoding be revealed consider a little longer message $U = aaab$ generated by the same source. The process of encoding is shown below, followed with step-by-step explanations.

$$\begin{aligned} &[0,\, 16) \xrightarrow{a} [0,\, 12) \xrightarrow{a} [0,\, 9) \xrightarrow{a} [0,\, 6 + 3/4)\\ &\qquad\qquad \to \begin{cases} [0,\, 6) & \text{with probability } 24/27\\ [6,\, 6 + 3/4) & \text{with probability } 3/27 \end{cases}\\ &[0,\, 6) \xrightarrow{0} [0,\, 12) \xrightarrow{b} [9,\, 12) \xrightarrow{10} [4,\, 16)\\ &\qquad\qquad \xrightarrow{\text{h.s.}} \begin{cases} [4,\, 8) \to 01 & \text{with probability } 1/3\\ [8,\, 16) \to 1 & \text{with probability } 2/3 \end{cases}\\ &[6,\, 6 + 3/4) \xrightarrow{0110} [0,\, 12) \to \cdots. \end{aligned}$$

The entire range is $[0,\, 2^4)$ since, for the given source, $\tau = 2$. The first letter $a$ narrows this interval to $3/4$ of its size in proportion to the share the letter $a$ has in the cumulative probability distribution, i.e., the interval $[0,\, 12)$ is a letter interval allocated to the letter $a$ within the entire range $[0,\, 16)$ (the remaining space $[12,\, 16)$ is allocated to $b$). The second and third letters continue narrowing a current interval similarly, obtaining letter intervals for digram $aa$ and trigram $aaa$. But the interval for $aaa$ has a noninteger right bound and we cannot proceed using only 4-bit integer arithmetic. So we split the interval into two homophonic intervals and select one of them for further narrowing. The sizes of homophonic intervals are in the proportion of $24/4$ to $3/4$ and their sum is $27/4$, hence the probabilities of selection.

Suppose that the interval $[0,\, 6)$ is picked. The next operation shown is scaling. Since $[0,\, 6)$ fits entirely within $[0,\, 8)$, i.e., within the left part of the entire range, and occupies $3/4$ of it, transmit $0$ to specify the left part and consider the left part as the entire range, scaling the interval to $[0,\, 12)$ to keep up proportions. The letter $b$ narrows the interval to $1/4$ of its size, $[9,\, 12)$ ($[0,\, 9)$ is allocated to $a$). Since $[9,\, 12)$ fits entirely within $[8,\, 12)$, i.e., within the third quarter of the entire range, and occupies $3/4$ of it, transmit $10$ to specify the third quarter and consider it as the entire range with the proportional scaling of the letter interval. The last operation is homophonic substitution for the final interval $[4,\, 16)$, which is done by partitioning it into homophone intervals.

To finish the example, return to the second alternative after splitting the interval. Suppose now that the less probable interval $[6,\, 6 + 3/4)$ is picked. Transmit the binary code for $6$ and consider the first $3/4$ of the entire range as a new interval. Further operations are similar to those explained above.

Tracing all transmitted bits plus bits obtained after homophonic substitution for the final interval enables us to construct all possible codewords for the message

$$aaab \to \begin{cases} 01001 & \text{with probability } 24/81\\ 0101 & \text{with probability } 48/81\\ 01101001 & \text{with probability } 3/81\\ 0110101 & \text{with probability } 6/81 \end{cases}$$

where each probability is obtained as a product of all choice probabilities for a particular codeword. Find, e.g., the probability of the codeword $01001$ to appear on the encoder output

$$P(01001) = P(aaab)P(01001|aaab) = \frac{27}{256} \cdot \frac{24}{81} = \frac{1}{32} = 2^{-5}$$

i.e., $P(01001)$ is equal to probability of a particular combination of five random bits. The similar results may be obtained for all other codewords.

In comparison with the BHC method, the ACIS method is more efficient from the viewpoint of computational complexity. But its random bit consumption is much greater than that of BHC method because, in general, splitting an interval may occur after processing each symbol of the message. Hence, the next idea is to combine the two methods in order to get benefits from low random bit consumption of the one and computational efficiency of the other. The principle of this combining is applying the ACIS method to blocks of symbols, the parameters $Q$ and $\hat{Q}$ for the blocks being computed by means of (a part of) the BHC method.

All the methods considered are described in subsequent sections. Being applied to a single symbol, the BHC and ACIS methods lead to the same letterwise coding, which, therefore, may be regarded as an underlying scheme. It is convenient to begin by describing the letterwise homophonic coding.

## III. Letterwise Homophonic Coding

In this section, we present an efficient implementation of interval partitioning, described in Section II, which is used for homophonic encoding and decoding of a single source letter. The key idea is that this partitioning can be done by examining bits of numbers representing the interval bounds and combined with homophone selection.

### A. Letterwise Encoding

To encode a symbol $u$ we need an interval $[Q(u),\, \hat{Q}(u))$ allocated to the symbol in the cumulative probability distribution. The encoding algorithm is the following.

Find binary representations for the interval bounds

$$\begin{aligned} Q(u) &= \vartheta(u)/\delta = 0.q_1q_2\cdots q_\tau q_{\tau+1}q_{\tau+2}\cdots \\ \hat{Q}(u) &= \hat{\vartheta}(u)/\delta = \hat{q}_0.\hat{q}_1\hat{q}_2\cdots \hat{q}_\tau\hat{q}_{\tau+1}\hat{q}_{\tau+2}\cdots \end{aligned}$$

(only the first $\tau$ bits are needed at once; further bits are obtained one by one when required).

Compute an auxiliary $\tau$-bit value $\tilde{Q}(u)$

$$\tilde{Q}(u) = \begin{cases} \Big(\lfloor 2^\tau \hat{Q}(u)\rfloor - 1\Big)/2^\tau, & \text{if } \hat{q}_i = 0 \text{ for all } i > \tau \\ \lfloor 2^\tau \hat{Q}(u)\rfloor/2^\tau, & \text{otherwise.} \end{cases}$$

Examining not more than $\tau$ successive bits of $Q(u)$ and $\tilde{Q}(u)$ find the following presentation:

$$\begin{aligned} Q(u) &= 0.b_1b_2\cdots b_c0\,11\cdots1\,q_{n+1}q_{n+2}\cdots q_\tau\cdots, \\ \tilde{Q}(u) &= 0.b_1b_2\cdots b_c1\underbrace{00\cdots0}_{s}\tilde{q}_{n+1}\tilde{q}_{n+2}\cdots\tilde{q}_\tau, \\ c &\geq 0, \quad s \geq 0, \quad n = c+s+1. \end{aligned} \tag{5}$$

In a special case when $s = 0$ and $q_i = \hat{q}_i = 0$ for all $i \geq n$, we obtain a codeword

$$C(u) = b_1b_2\cdots b_c.$$

Otherwise, construct a codeword as follows:

$$C(u) = b_1b_2\cdots b_c r_0 e_1e_2\cdots e_s r_1r_2\cdots r_k \tag{6}$$

where $e_1 = e_2 = \cdots = e_s = 1 - r_0$ and $r_0r_1r_2\cdots r_k$ is a random bit sequence of minimal length that makes $C(u)$, viewed as a binary fraction $0.C(u)$ expanded by an arbitrary continuation $*$, satisfy the inequality

$$Q(u) \leq 0.C(u)* < \hat{Q}(u). \tag{7}$$

To determine whether a particular random bit sequence meets the requirements imposed by (7), we need to test the following conditions. If $r_0 = 0$ and $q_i = 0$ for all $i > n$ or $r_0 = 1$ and $\hat{q}_i = 0$ for all $i > n$ then no other random bits are needed, $k = 0$. Otherwise, get extra random bits $r_1, r_2, \cdots$, until such bit $r_k$ is encountered that one of the following conditions hold:

$$r_k > q_{n+k} \text{ or } (r_k = q_{n+k} \text{ and } q_i = 0 \text{ for all } i > n+k) \qquad (r_0 = 0) \tag{8}$$

$$r_k < \hat{q}_{n+k} \qquad (r_0 = 1) \tag{9}$$

$$r_k < q_{n+k} \qquad (r_0 = 0) \tag{10}$$

$$r_k > \hat{q}_{n+k} \text{ or } (r_k = \hat{q}_{n+k} \text{ and } \hat{q}_i = 0 \text{ for all } i > n+k) \qquad (r_0 = 1). \tag{11}$$

If either (8) or (9) is true then (7) is satisfied and the construction of the code is completed. If (10) or (11) holds then the whole sequence $r_0r_1\cdots r_k$ must be rejected since it leads out of the interval and the process of finding a relevant random bit sequence must be rerun once again.

### B. Letterwise Decoding

In decoding, a question arises how many code bits are to be taken into consideration since the actual length of a codeword is unknown. Let the decoder input receive a code sequence $c_1c_2c_3\cdots$, where $c_i$ denotes the $i$th code bit. Consider the sequence as an infinite binary fraction

$$C = 0.c_1c_2\cdots c_\tau c_{\tau+1}c_{\tau+2}\cdots$$

(continued with zeros if necessary). A codeword built for the interval $[\vartheta/\delta,\, \hat{\vartheta}/\delta)$ and, perhaps, supplemented by some random continuation, specifies a point $\Theta/\delta$ belonging to the interval. We have

$$\Theta = \lfloor C\delta \rfloor.$$

Denote by $C_\tau$ the binary fraction formed by the first $\tau$ code bits

$$C_\tau = 0.c_1c_2\cdots c_\tau.$$

It can be easily seen that, since $\delta \leq 2^\tau$, $\lfloor C_\tau\delta\rfloor \leq \lfloor C\delta\rfloor \leq \lfloor C_\tau\delta\rfloor + 1$, and sequential multiplying $\delta$ by $c_{\tau+1}, c_{\tau+2}, \cdots$, is to be done until the carry into the integer part arises or the product gets a zero bit in the fractional part (which implies that the carry is no longer possible). Hence we suggest the following algorithm to compute $\Theta$.

a) Compute the product $C_\tau\delta$. Let $X = C_\tau\delta - \lfloor C_\tau\delta\rfloor$, $X < 1$. Set $i = \tau$.

b) Perform the following operations:

$$\begin{aligned} &X := X \times 2, \quad \alpha := \lfloor X \rfloor, \quad X := X - \alpha; \\ &i := i+1, \quad \text{if } c_i = 1 \text{ then } X := X + \delta/2^\tau; \\ &\beta := \lfloor X \rfloor, \quad X := X - \beta. \end{aligned}$$

c) If $\alpha = \beta$ then $\Theta = \lfloor C_\tau\delta\rfloor + \alpha$; else go to b).

As $\Theta$ is known, find the letter $u$ satisfying the inequality

$$\vartheta(u) \leq \Theta < \hat{\vartheta}(u).$$

This is the encoded symbol. To finish decoding, reconstruct a codeword for the letter $u$ using correspondent code bits instead of random ones, and extract the codeword from the code sequence.

Let us consider an example. Let $A = \{a, b\}$, $P(a) = 2/3$, $P(b) = 1/3$, and $\tau = 2$. Suppose we need to encode the letter $b$ provided that the sequence of random bits is $00\,011\cdots$. The interval allocated to the letter $b$ is $[2/3,\, 1)$, so we have

$$\begin{aligned} Q(b) &= 0.101010\cdots \\ \hat{Q}(b) &= 1.0000\cdots \\ \tilde{Q}(b) &= 0.11. \end{aligned}$$

By examining two bits of $Q(b)$ and $\tilde{Q}(b)$ we obtain

$$c = 1, \quad s = 0, \quad n = 2$$

and a codeword is to be built in the form

$$C(b) = 1r_0r_1\cdots r_k.$$

The first two random bits give $r_0 = 0$, $r_1 = 0$, and are to be rejected since $r_1 < q_3$ (see (10)). The further random bits give $r_0 = 0$, $r_1 = 1 = q_3$, $r_2 = 1 > q_4$ ((8) is satisfied for $r_2$) and we obtain the codeword

$$C(b) = 1011.$$

In decoding, we have the code sequence $c_1c_2c_3c_4* = 1011*$, where $*$ is an arbitrary binary continuation (produced by other codewords). For our source, $\tau = 2$ but the first two code bits do not allow us to decide on the letter encoded since both letters $a$ and

$b$ may have codewords that begin with $10$. Suppose, for simplicity, that $* = 000\cdots$, thus permitting the maximal "trend" of the code sequence toward the letter $a$. We have

$$C_\tau = 0.10 \qquad C_\tau\delta = 1.10 \qquad X = 0.10.$$

Perform Step b)

$$\begin{aligned} \alpha &= 1, \quad X = 0.0;\\ c_3 &= 1 \rightsquigarrow X = 0.11;\\ \beta &= 0, \quad X = 0.11. \end{aligned}$$

Since $\alpha \neq \beta$ repeat Step b)

$$\alpha = 1, \quad X = 0.1; \qquad c_4 = 1 \rightsquigarrow X = 1.01; \qquad \beta = 1.$$

Now $\alpha = \beta$ and we obtain $\Theta = 1 + 1 = 2$. The encoded letter is $b$ since $\vartheta(b) = 2$. In this particular example we detected all four bits of the codeword. However, in the general case, the number of code bits can be less than or equal to $\tau$, so we need to reconstruct a codeword after finding the encoded symbol. In our example, we could do this by encoding the letter $b$ using $c_2c_3c_4$ instead of $r_0r_1r_2$.

### C. Letterwise Coding Properties

*Lemma 1:* In computation of $\Theta$, the average number of repetitions of Step b) does not exceed 2.

*Proof:* Assume that the code sequence is completely random. Then $C\delta$ is a random variable uniformly distributed in $[0, \delta)$ and its binary representation (apart from the first $\tau$ bits) may be treated as a random bit sequence. Consider only one of the conditions to stop iterations, namely, $\alpha = \beta = 0$. It means that a 0 bit appears in the binary representation of $C\delta$, probability of this event being $1/2$. Hence, two efforts are required on average to get $\alpha = \beta = 0$ satisfied.

*Theorem 1:* Let the described letterwise homophonic coding be applied to messages generated by a Bernoulli source with known statistics. Then the following propositions hold:

i) to construct a codeword at most 12 random bits are needed on average;

ii) the mean codeword length does not exceed $H(u) + 3$ bits, where $H(u)$ is the entropy of a source letter;

iii) the mean time of encoding and decoding is determined by several $\tau$-bit multiplication-type operations (two divisions in encoding, one multiplication and two divisions in decoding).

*Proof:* To prove Proposition i), first, find the mean length $N_r$ of the random bit sequence that satisfies one of the conditions (8)–(11). Notice that one bit $(r_0)$ is always needed. To provide an upper estimate assume that $\tau = \infty$ and neither $Q(u)$ nor $\hat{Q}(u)$ contain tails of zeros. Then the number of additional random bits will be determined by how long the equality between an immediate bit $r_k$ and a corresponding bit $q_{n+k}$ (or $\hat{q}_{n+k}$) is held, the probability of this equality being

$$\begin{aligned} &\Pr\{r_k = q_{n+k}\}\\ &\quad = \Pr\{r_k = 0\} \times \Pr\{q_{n+k} = 0\}\\ &\qquad + \Pr\{r_k = 1\} \times (1 - \Pr\{q_{n+k} = 0\}) = 1/2. \end{aligned}$$

This means that, apart from $r_0$, one more random bit will be needed with probability $1/2$, after which one more random bit will again be needed with the same probability, and so on. Hence

$$N_r = 1 + \frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{8} \times 3 + \cdots = 1 + \sum_{n=1}^{\infty} \frac{n}{2^n} = 3.$$

Since either $q_{n+1} \neq 1$ or $\hat{q}_{n+1} \neq 0$ (see (5)) any random bit sequence will at worst satisfy either (8) or (9) with probability $1/4$. Consequently, there will be required four different sequences on average and the total mean number of random bits will be $4N_r = 12$.

For Proposition ii), let the codeword constructed for the symbol $u \in A$ be defined as (6). Its length equals $c+s+1+k$, $1+k$ being the number of random bits. Denote by $l(u)$ the mean codeword length for the letter $u$

$$l(u) = c + s + N_r.$$

Since the size of the interval equals $P(u)$, $c + s \leq \lfloor -\log P(u) \rfloor$. Hence

$$l(u) \leq \lfloor -\log P(u) \rfloor + 3 \leq -\log P(u) + 3.$$

The expected value of $l(u)$ over the whole set of $u \in A$

$$\begin{aligned} E[l(u)] &= \sum_{u\in A} P(u)l(u) \leq \sum_{u\in A} P(u)(-\log P(u) + 3)\\ &= -\sum_{u\in A} P(u)\log P(u) + 3\sum_{u\in A} P(u) = H(u) + 3. \end{aligned}$$

Finally, taking into account the description of the algorithm, Lemma 1, and the first proposition of the theorem, Proposition iii) becomes obvious. □

## IV. Block Homophonic Coding

Let there be given a message (or a block of symbols) $U$ and, for ease of designation, the length $L$ of the message be a power of 2, $L = 2^\sigma$. In [10], a fast and space-efficient algorithm to compute $Q(U)$ and $\hat{Q}(U)$ has been proposed. For the purpose of randomization we accomodate this algorithm to rational symbol probabilities. Let the individual letter and cumulative probabilities be

$$\begin{aligned} &P(u_1) = \rho_1^0/\delta_0, \cdots, P(u_L) = \rho_L^0/\delta_0\\ &Q(u_1) = \vartheta_1^0/\delta_0, \cdots, Q(u_L) = \vartheta_L^0/\delta_0, \qquad L = 2^\sigma. \end{aligned}$$

### A. BHC Encoding

Compute

$$\begin{aligned} \vartheta_k^i &= \vartheta_{2k-1}^{i-1}\delta_{i-1} + \rho_{2k-1}^{i-1}\vartheta_{2k}^{i-1}\\ \rho_k^i &= \rho_{2k-1}^{i-1}\rho_{2k}^{i-1}, \qquad k = 1, 2, \cdots, L/2^i\\ \delta_i &= (\delta_{i-1})^2, \qquad\quad i = 1, 2, \cdots, \sigma. \end{aligned} \tag{12}$$

Then $Q(U) = \vartheta_1^\sigma/\delta_\sigma$, $\hat{Q}(U) = (\rho_1^\sigma + \vartheta_1^\sigma)/\delta_\sigma$.

Note that $Q(U)$ and $\hat{Q}(U)$ are ratios of $L\tau$-bit numbers. To find homophonic code for $U$ apply to the interval $[Q(U), \hat{Q}(U))$ the letterwise homophonic encoding method described in Section III with only one modification: replace everywhere $\tau$ by $L\tau$.

### B. BHC Decoding

Let the decoder input receive a code sequence $C$. On the basis of at least $L\tau$ code bits compute $\Theta = \lfloor C\delta_\sigma \rfloor$ (see Section III).

Let $Z_1^\sigma = \Theta$. Compute

$$Z_1^{\sigma-1} = \lfloor Z_1^\sigma/\delta_{\sigma-1} \rfloor, \cdots, Z_1^1 = \lfloor Z_1^2/\delta_1 \rfloor, Z_1^0 = \lfloor Z_1^1/\delta_0 \rfloor.$$

Find the symbol $u_1$ that satisfies the inequality $\vartheta(u_1) \leq Z_1^0 < \hat{\vartheta}(u_1)$. It will be the first encoded symbol. Compute $Z_2^0 = \lfloor (Z_1^1 - \vartheta_1^0\delta_0)/\rho_1^0 \rfloor$ and find $u_2$ satisfying $\vartheta(u_2) \leq Z_2^0 < \hat{\vartheta}(u_2)$. Using (12) compute $\vartheta_1^1$ and $\rho_1^1$. Compute $Z_2^1 = \lfloor (Z_1^2 - \vartheta_1^1\delta_1)/\rho_1^1 \rfloor$ and from $Z_3^0 = \lfloor Z_2^1/\delta_0 \rfloor$ find the symbol $u_3$, and from $Z_4^0 = \lfloor (Z_2^1 - \vartheta_3^0\delta_0)/\rho_3^0 \rfloor$ find $u_4$. As we have known the first four symbols, obtain $\vartheta_1^2$ and $\rho_1^2$. From $Z_2^2 = \lfloor (Z_1^3 - \vartheta_1^2\delta_2)/\rho_1^2 \rfloor$, proceeding in the same manner, find $u_5$, $u_6$, $u_7$, $u_8$, $\vartheta_2^3$, and $\rho_2^3$. The process going on, we shall eventually obtain

$$Z_2^{\sigma-1} = \lfloor (Z_1^\sigma - \vartheta_1^{\sigma-1}\delta_{\sigma-1})/\rho_1^{\sigma-1} \rfloor$$

that will give us the last $L/2$ encoded symbols, as well as $\vartheta_2^{\sigma-1}$ and $\rho_2^{\sigma-1}$. After having decoded the whole message, compute $\vartheta_1^{\sigma}$ and $\rho_1^{\sigma}$ and reconstruct the codeword using corresponding input bits instead of random ones. Extracting the codeword from the input sequence completes the decoding.

### C. BHC Method Properties

*Theorem 2:* Let the BHC algorithm be applied to encoding messages of the length $L$, $L \geq 1$, generated by a Bernoulli source with known statistics. Then the following propositions hold:

i) the mean per symbol redundancy $r$ does not exceed $3/L$ bits and not more than $12/L = 4r$ random bits per symbol are required on average;

ii) the memory size of the encoder and decoder increases as $O(L)$, and the mean per symbol time of encoding and decoding grows as $O(\log^2 L \log\log L)$ as $L \to \infty$.

*Proof:* The first proposition immediately follows from Theorem 1.

The estimate of the memory size required is based on the observation that in (12), for every layer $i$ formed by all $\vartheta^i$ and $\rho^i$, the memory amount of $2L\tau$ bits is needed and all values of the $i$th layer depend only on those of the $(i-1)$th layer, so only two layers are to be stored at any instant. In decoding, the additional memory for $Z$ values is also confined to $2L\tau$ bits, since at any instant only one value of every layer suffices to be stored.

The estimate of the algorithm's time consumption is obtained by summing up complexities of the operations involved. We assume that Schönhage–Strassen's method should be used for multiplication and division (see [11]). This method requires $O(n \log n \log\log n)$ bit operations for multiplying two $n$-bit numbers or dividing $2n$-bit number by $n$-bit one. According to [10], it causes the complexity of computing $\rho_1^{\sigma}$, $\vartheta_1^{\sigma}$, and $Z_k^i$ to be determined as $O(L \log^2 L \log\log L)$, as $L \to \infty$, while the letterwise homophonic coding, given $Q(U)$ and $\hat{Q}(U)$, requires $O(L \log L \log\log L)$ operations (see Lemma 1), from which the proposition of the theorem immediately follows. □

*Corollary 1:* The estimates of the memory size and mean-per-symbol time considered as functions of the redundancy $r$, are determined as $O(1/r)$ and $O(\log^2 1/r \log\log 1/r)$, respectively, as $r \to 0$.

## V. Arithmetic Coding with Interval Splitting

Denote by $l$ and $h$ the lower and higher bounds of the interval. Let $l$ and $h$ be $t$-bit unsigned integers (for ease of designation, allow $h$ to take the value $2^t$). We claim that $t$ satisfy the inequality

$$t \geq \tau + 2 \tag{13}$$

which will guarantee that no symbol maps into the interval of the size less than one unit.

### A. ACIS Encoding

First define the scaling operation for the interval $[l, h)$. Consider the closed interval $[l, g]$, where $g = h - 1 + 0.111\cdots$ (the number $0.111\cdots$ is in binary notation). Let $l$ and $g$ have binary expansions

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

By examining bits of $l$ and $g$ find the following presentation:

$$\begin{aligned} l &= b_1 b_2 \cdots b_c 0\, 11 \cdots 1\, l_{n+1} \cdots l_t.000\cdots, \\ g &= b_1 b_2 \cdots b_c 1\, \underbrace{00\cdots 0}_{s} g_{n+1} \cdots g_t.111\cdots, \\ & 0 \leq c \leq t, \quad 0 \leq s \leq t-1, \quad n = c+s+1 \end{aligned} \tag{14}$$

(here the upper bounds for $c$ and $s$ are due to (13)). The scaling, then, results in transmitting $b_1 b_2 \cdots b_c$ to the encoder output, incrementing the "squeeze counter" $S$ by $s$, and computing new interval bounds

$$\begin{aligned} l' &= 0 l_{n+1} \cdots l_t 0 \cdots 0.000\cdots \\ g' &= 1 g_{n+1} \cdots g_t 1 \cdots 1.111\cdots. \end{aligned} \tag{15}$$

Squeezing an interval will entail transmitting $S$ zeros or ones after the next code bit is determined. Therefore, transmitting $b_1$ must be followed by transmitting $S$ opposite bits $1 - b_1$ (with consequent setting $S$ to zero) due to squeezing in the previous scaling operation(s). We shall call the interval $[l, h)$, where $l = l'$ and $h = \lfloor g' \rfloor + 1$, normalized.

Notice that if for a large number of successive symbols of $U$ scaling results only in increasing $S$ then $S$ may overflow. Since for each symbol $u_1, u_2, \cdots, u_L$, $S$ may be increased by $t-1$ at most (this is the maximal value of $s$ in (14)), to avoid overflow $L(t-1) < 2^{t_S}$ must hold, where $t_S$ is the size of $S$ in bits. Hence we have a restriction on the message length

$$L < \frac{2^{t_S}}{t-1}. \tag{16}$$

Initially, the interval $[l, h)$ is the entire range $[0, 2^t)$ and $S$ is set to zero. Denote by $d$ the size of the interval $d = h - l$.

For every symbol $u \in U$ compute the interval within $[l, h)$ corresponding to the symbol $u$

$$\left[l + dQ(u),\, l + d\hat{Q}(u)\right) = [I_l + R_l/\delta,\, I_h + R_h/\delta)$$

where $I_l$ and $R_l$ denote the integer part and the remainder for the value $l + dQ(u) = l + d\vartheta(u)/\delta$, $I_h$ and $R_h$ denote the same for $l + d\hat{Q}(u)$.

Split the obtained interval into three homophonic intervals $V^-$, $V$, and $V^+$

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

and choose one of them with probability proportional to its size. It is convenient to represent sizes of the intervals by integer numbers $v^-$, $v$, and $v^+$ proportional to the real sizes

$$\begin{aligned} v^- &= (\delta - R_l) \bmod \delta \\ v^+ &= R_h \\ v &= d\hat{\vartheta}(u) - d\vartheta(u) - v^+ - v^-. \end{aligned}$$

The problem of probabilistic selection will be discussed later.

If the interval $V$ with integer bounds is picked then set $l = I_l + \lceil R_l/\delta \rceil$, $h = I_h$, and perform scaling for the interval $[l, h)$. The encoding of the symbol $u$ is terminated.

If the interval $V^+$ is chosen then scale the embracing unit-sized interval $[I_h, I_h + 1)$ into the entire range and compute the new bounds for $V^+$

$$V^+ = \left[0,\, I'_h + R'_h/\delta\right)$$

where $I'_h = \lfloor 2^t R_h/\delta \rfloor$ and $R'_h = (2^t R_h) \bmod \delta$. Set $I_l = R_l = 0$, $I_h = I'_h$, $R_h = R'_h$ and apply recursively the above algorithm of splitting an interval.

Similarly, if the interval $V^-$ is chosen then scale interval $[I_l, I_l + 1)$ and compute the new bounds for $V^-$

$$V^- = \left[I'_l + R'_l/\delta,\, 2^t\right)$$

where $I'_l = \lfloor 2^t R_l/\delta \rfloor$ and $R'_l = (2^t R_l) \bmod \delta$. Set $I_l = I'_l$, $R_l = R'_l$, $I_h = 2^t$, $R_h = 0$ and continue splitting an interval.

After the last symbol of the message is processed we have the final normalized interval $[l, h)$ and the counter $S$. If the final interval is not $[0, 2^t)$ or $S \neq 0$ then, for perfect randomization, we need to perform a homophonic substitution for the interval. Find a random bit sequence $r_0 r_1 \cdots r_k$ that fits within the interval just like we do in letterwise homophonic encoding. Transmit $r_0 r_1 \cdots r_k$ to the encoder output, $r_0$ being followed by $S$ opposite bits $(1 - r_0)$. The encoding of the message is finished.

### *B. ACIS Decoding*

Let the decoder input receive a code sequence. Introduce a variable $c$ to contain $t$ current code bits.

The scaling operation for decoding is similar to that for encoding except for the following. Neither transmitting bits nor maintaining squeeze counter are needed. Instead, as bits are shifted out from $l$ and $g$, the same-order bits are to be shifted out from $c$, new code bits being shifted in.

Initially, $c$ contains the first $t$ code bits and the interval $[l, h)$ is the entire range $[0, 2^t)$.

To determine each encoded symbol, it is necessary to find which letter of the alphabet $A$ might reduce the interval $[l, h)$ so that $c$ lies within the reduced interval. To do that compute $I_c = \lfloor (c - l)\delta/d \rfloor$ and $R_c = ((c - l)\delta) \bmod d$ and find the symbol $u$ satisfying the inequality $\vartheta(u) \leq I_c < \hat{\vartheta}(u)$. But if

$$I_c = \hat{\vartheta}(u) - 1 \text{ and } R_c > d - \delta \tag{17}$$

then taking into account extra code bits can add 1 to $I_c$ making it equal to $\vartheta(w) = \hat{\vartheta}(u)$, where $w$ is the next letter after $u$ in the alphabet $A$. This happens only if the homophonic interval $V^+$ has been chosen for the letter $u$, or $V^-$ has been chosen for the letter $w$ in encoding. To distinguish between these two letters consider the alphabet $B = \{u, w\}$ with cumulative probability distribution

$$Q(u) = 0 \qquad \hat{Q}(u) = Q(w) = (d - R_c)/\delta \qquad \hat{Q}(w) = 1$$

take the next $t$ code bits in $c$, changing the interval $[l, h)$ to $[0, 2^t)$, and apply recursively the decoding algorithm to the alphabet $B$.

If the condition (17) does not hold then $u$ is the encoded symbol. Reduce the interval $[l, h)$ to $[l + \lceil dQ(u) \rceil, \; l + \lfloor d\hat{Q}(u) \rfloor)$ and perform scaling. The decoding of the symbol $u$ is finished.

### *C. ACIS Method Properties*

*Lemma 2:* The size $d$ of the normalized interval $[l, h)$ is at least $2^{t-2} + 2$.

*Proof:* Consider a closed interval $[l, g]$, $g = h - 1 + 0,\, 111 \cdots$. Then its size

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

There are three variants of normalized intervals possible: $[00{*}, 11{*}]$, $[00{*}, 10{*}]$, and $[01{*}, 11{*}]$, where $*$ denotes $t - 2$ arbitrary binary digits. The size of the interval in the first case is greater than that in the second case, the second and the third cases being equivalent. For the second case we have

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

Hence

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

*Theorem 3:* Let there be given a message $U = u_1 u_2 \cdots u_L$ generated by a source over the alphabet $A = \{a_1, a_2, \cdots, a_N\}$ and probability distribution $P(a_1), P(a_2), \cdots, P(a_N)$ for every symbol $u \in U$. Apply to the message $U$ the ACIS method, the interval bounds being represented by $t$ bit numbers, $t \geq 4$. Then the following propositions hold.

i) The redundancy $r$ per symbol $u$ satisfies the inequality

$$r(u) < 8Nt2^{-t} + 3/L. \tag{18}$$

ii) The memory size $M$ of the encoder (decoder) and the time $T$ of encoding (decoding), seen as functions of $t$, $t \to \infty$, are determined by the estimates

$$\begin{aligned} M &= O(t) \\ T &= O(t \log t \log\log\, t). \end{aligned}$$

*Proof:* Suppose that the message length is infinite $(L = \infty)$. Denote by $E[C(u)]$ the mean codeword length for a symbol $u$. Denote by $H(u)$ the entropy of a source letter. In the ACIS encoding every source symbol $u$ is virtually represented by homophones $v, v_1^-, v_1^+, v_2^-, v_2^+, \cdots$, picked with probabilities $P(v), P(v_1^-), P(v_1^+), \cdots$, the sum of which is $P(u)$. Then

$$E[C(u)] = \sum_{u \in A} e(u)$$

where

$$\begin{aligned} e(u) = -\big[&P(v)\log P(v) + P(v_1^-)\log P(v_1^-) \\ &+ P(v_1^+)\log P(v_1^+) + \cdots\big]. \end{aligned} \tag{19}$$

Since homophones $v_1^-$ and $v_1^+$ occur due to normalized interval splitting and the size of the normalized interval is greater than $2^{t-2}$ (by Lemma 2), probabilities $P(v_1^-)$ and $P(v_1^+)$ cannot exceed $2^{-(t-2)}$. Further homophones $v_2^-$ and $v_2^+$ occur due to splitting the entire range $[0, 2^t)$ and their probabilities do not exceed $2^{-(t-2)} \cdot 2^{-t}$, etc. Therefore,

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

Since

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

it can be easily shown that

$$P(u)\log P(u) - P(v)\log P(v) < \frac{1}{\ln 2}\big(1 - 8 \cdot 2^{-t}\big)8 \cdot 2^{-t}.$$

Then

$$\begin{aligned} r(u) &= E[C(u)] - H(u) \\ &= \sum_{i=1}^{N} e(a_i) - \sum_{i=1}^{N} -P(a_i)\log P(a_i) \\ &< \sum_{i=1}^{N} \bigg(P(a_i)\log P(a_i) - P(v)\log P(v) \\ &\qquad + 8t\frac{2^{-t}}{(1-2^{-t})^2} - 16\frac{2^{-t}}{1-2^{-t}}\bigg) \\ &< \sum_{i=1}^{N} 8t2^{-t}\left[\frac{1}{t\ln 2} + \frac{1}{(1-2^{-t})^2} - \frac{2}{t(1-2^{-t})}\right] \\ &< \sum_{i=1}^{N} 8t2^{-t} = 8Nt2^{-t}. \end{aligned}$$

If the message length $L$ is finite then, according to Theorem 1, homophonic encoding of the final interval adds to the code sequence 3 bits on average and the first proposition of the theorem is proved.

The second proposition is based on the observation that we must store $l$, $h$, and several auxilliary $t$ bit variables while the rest of data have sizes independent of $t$. The estimate of the computation time is determined by asymptotic complexity of $t$ bit multiplication and division (see Schönhage–Strassen algorithm in [11]). □

*Corollary 2:* The memory size $M$ and the time $T$, seen as functions of $r$, $r \to 0$, are determined by the estimates

$$\begin{aligned} M &= O(\log 1/r) \\ T &= O(\log 1/r \log\log\ 1/r \log\log\log 1/r). \end{aligned} \tag{20}$$

### D. Realization of Interval Selection

To pick one interval from three or two homophonic intervals with probability proportional to its size we suggest a simple (but obviously not optimal with respect to the number of random bits) method.

Let there be given three intervals $A, B, C$ with sizes $a, b, c$. Assume that they are ordered by size: $a \geq c \geq b$. Consider the interval $Z$ to be a concatenation of $A, B, C$, $z = a + b + c$ (note that the shortest interval lies between longer ones).

Divide the interval $Z$ into two equal parts and mark either the left or the right part according to a random bit value. If the marked part fits entirely within $A$, $B$, or $C$ then the corresponding interval is chosen. Otherwise, contract $Z$ to its marked part and repeat the process from the beginning.

It is obvious that the probability of a choice will be proportional to the size of the interval. Increasing arithmetic precision with each subsequent division may be avoided by scaling the marked part of $Z$ to its full size together with doubling the remaining parts of $A, B, C$.

A question arises how many random bits we need to make a choice. The following two lemmas can be easily proved.

*Lemma 3:* For making a choice between two intervals no more than two random bits are required on average.

*Lemma 4:* To make a choice between three intervals no more than three random bits are required on average.

In the encoding method we must choose between intervals $V$, $V^-$, and $V^+$. The size of $V$ is usually much greater than the sizes of $V^-$ and $V^+$. Besides, the problem of choice may arise many times for a symbol being encoded. A more accurate estimate of the number of random bits consumed is given by the following.

*Theorem 4:* For the ACIS encoding there exists a method of choosing a homophonic interval for which the average number of random bits $N_r$ used per source symbol $u$ satisfies the inequality

$$N_r(u) < 2 + r(u) + 9/L$$

where $r(u)$ is the code redundancy, $r(u) \to 0$, and $L$ is the message length.

*Proof:* For any interval sizes $a, b, c$ there exists an integer $k$ such that

$$(2^k - 1)(b + c) \leq a < (2^{k+1} - 1)(b + c). \tag{21}$$

Denote by $N_r^{(3)}$ the average number of random bits to pick the interval $A$, $B$, or $C$

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

In the ACIS encoding, homophonic intervals $V$, $V^-$, and $V^+$ take the part of the intervals $A$, $B$, and $C$, respectively. So we have

$$z = d\rho(u) \qquad b + c < 2\delta \qquad d > 2^{t-2} \tag{22}$$

(using Lemma 2 for the last inequality). Combining (22) and the right part of (21) obtain

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

Hence

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

and the expected value over the alphabet $A$

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

Splitting an interval may be invoked many times. For the first time, $N$ source letters are distributed over the normalized interval and it is necessary to pick one of three homophonic intervals. Subsequently, one of two intervals is to be picked within the entire range. Furthermore, up to 12 random bits on average are to be used for homophonic encoding of the final interval according to Theorem 1. Therefore,

$$\begin{aligned} N_r &< E\left[N_r^{(3)}(u)\right] + (N-1)2^{-(t-2)}\bigl(2 + 2^{-t}(2 + \cdots)\bigr) + 12/L \\ &< 2 + 24N2^{-t} + 12/L. \end{aligned}$$

Taking into account the estimate of redundancy (18) completes the proof. □

## VI. A General Method

The present method introduces a class of codes, based on a certain combination of the methods described earlier, to solve the coding problem when both redundancy and a number of random bits are specified. To meet these requirements we suggest to encode blocks of symbols by means of the ACIS method. If the size of a block is $m$ then about $2/m$ random bits are expected to be used per one symbol.

Denote by $Y = u_1 u_2 \cdots u_m$ a block of $m$ symbols. For ease of designation assume that $m$ is a power of 2 and the message length $L$ is a multiple of $m$, $m = 2^\sigma$, $L \bmod m = 0$. The message $U$ may be represented as $U = Y_1 Y_2 \cdots Y_{L/m}$, where $Y$ is a letter of the alphabet $A^m$. Define $Q(Y)$, and $\hat{Q}(Y)$ in the same manner as for $U$.

Let the individual letter and cumulative probabilities be

$$\begin{aligned} &P(u_1) = \rho_1^0/\delta_0, \cdots, P(u_m) = \rho_m^0/\delta_0 \\ &Q(u_1) = \vartheta_1^0/\delta_0, \cdots, Q(u_m) = \vartheta_m^0/\delta_0, \qquad m = 2^\sigma. \end{aligned}$$

Compute

$$\begin{aligned} \vartheta_k^i &= \vartheta_{2k-1}^{i-1}\delta_{i-1} + \rho_{2k-1}^{i-1}\vartheta_{2k}^{i-1} \\ \rho_k^i &= \rho_{2k-1}^{i-1}\rho_{2k}^{i-1}, \qquad k = 1, 2, \cdots, m/2^i \\ \delta_i &= (\delta_{i-1})^2, \qquad i = 1, 2, \cdots, \sigma. \end{aligned}$$

Then $Q(Y) = \vartheta_1^\sigma/\delta_\sigma$, $\hat{Q}(Y) = (\rho_1^\sigma + \vartheta_1^\sigma)/\delta_\sigma$, and we may apply the ACIS method to encode and decode $Y$. Note only that in decoding, taking the next letter of the alphabet, should be replaced by taking the lexicographically next block, the task being apparently solved in $O(m)$ time.

According to Theorem 3, the redundancy per block $r(Y) < 8N^m t2^{-t} + 3m/L$. Hence, the redundancy per letter

$$r(u) = \frac{r(Y)}{m} < \frac{8N^m t2^{-t}}{m} + \frac{3}{L}.$$

And by Theorem 4, the average number of random bits

$$N_r(u) = \frac{N_r(Y)}{m} < \frac{2 + r(Y) + 9m/L}{m} = \frac{2}{m} + r(u) + \frac{9}{L}.$$

To meet specified upper bounds for both redundancy $(r^*(u))$ and a number of random bits $(N_r^*(u))$, first, take $m$ satisfying the inequality

$$2/m \leq N_r^*(u) - r^*(u) - 9/L$$

and, next, take $t$ such that

$$t2^{-t} \leq \frac{m(r^*(u) - 3/L)}{8N^m}.$$

Notice that if random bit consumption is not restricted then, to obtain specified redundancy, the pure ACIS method is to be used and estimates of the memory size and computation time are given by (3). If random bit consumption is restricted to be as low as possible ($4r$ bits, $r \to 0$) then we have to use the BHC method whose estimates of the memory size and computation time are given by (2). If $4r < N_r^* < 2 + r$ then the block version of the ACIS method is to be applied whose complexity is intermediate between (2) and (3).

From the above consideration we can derive the following theorem.

*Theorem 5:* If the redundancy $r$ and the number of random bits $N_r$ are tied together by the equation $N_r = cr$, where $c$ may be a constant or a function of $r$, then the following asymptotic estimates of the memory size $M$ and computation time $T$ are valid:

$$M = O\left(\frac{1}{N_r} + \log \frac{N_r}{r}\right)$$

and

$$T = O\left(\log^2 \frac{1}{N_r} \log\log \frac{1}{N_r} + \log \frac{N_r}{r} \log\left(\frac{1}{N_r} + \log \frac{N_r}{r}\right) \times \log\log\left(\frac{1}{N_r} + \log \frac{N_r}{r}\right)\right)$$

as $r \to 0$.

Note that when $c$ is constant, i.e., $N_r$ is proportional to $r$, we obtain the estimates (2). If $c$ is in reverse proportion to $r$, i.e., $N_r$ is constant, the estimates (3) are obtained.

## REFERENCES

[1] C. Shannon, "Communication theory of secrecy systems," *Bell Syst. Tech. J.*, vol. 28, no. 4, pp. 656–715, 1949.

[2] Ch. G. Günther, "A universal algorithm for homophonic coding," in *Advances in Cryptology Eurocrypt-88* (Lecture Notes in Computer Science, vol. 330). Heidelberg/New York: Springer-Verlag, 1988, pp. 405–414.

[3] H. N. Jendal, Y. J. B. Kuhn, and J. L. Massey, "An information-theoretic treatment of homophonic substitution," in *Advances in Cryptology Eurocrypt-89* (Lecture Notes in Computer Science, vol. 434). Berlin: Springer-Verlag, 1990, pp. 382–394.

[4] V. C. da Rocha Jr. and J. L. Massey, "On the entropy bound for optimum homophonic substitution," in *Proc. IEEE Int. Symp. Inform. Theory* (Ulm, Germany, July 1997), p. 93.

[5] B. Y. Ryabko and A. N. Fionov, "A fast and efficient homophonic coding algorithm," in *Algorithms and Computation ISAAC-96* (Lecture Notes in Computer Science, vol. 1178). Berlin, Germay: Springer-Verlag, 1996, pp. 427–438.

[6] ——, "Decreasing redundancy of homophonic coding," in *Proc. IEEE Int. Symp. Inform. Theory* (Ulm, Germany, July 1997), p. 94.

[7] J. Rissanen and G. G. Langdon, "Universal modeling and coding," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 12–23, Jan. 1981.

[8] T. C. Bell, J. G. Cleary, and I. H. Witten, *Text Compression.* Englewood Cliffs, NJ: Prentice-Hall, 1990.

[9] W. Penzhorn, "A fast homophonic coding algorithm based on arithmetic coding," in *Fast Software Encryption* (Lecture Notes in Computer Science, vol. 1008). Berlin, Germany: Springer-Verlag, 1995, pp. 329–345.

[10] B. Y. Ryabko, "Fast and effective coding of information sources," *IEEE Trans. Inform. Theory*, vol. 40, pp. 96–99, Jan. 1994.

[11] A. V. Aho, L. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms.* Reading, MA: Addison-Wesley, 1976.

# Strongly Universal Hashing and Identification Codes via Channels

Kaoru Kurosawa, *Member IEEE*, and Takuya Yoshida

***Abstract*—This correspondence shows that $\epsilon$-almost strongly universal classes of hash functions can yield better explicit constructions of identification codes via channels (ID codes) and identification plus transmission codes (IT codes) than the previous explicit constructions of Verdú and Wei.**

***Index Terms*—Binary constant-weight code, explicit construction, identification code via channels, universal hash function.**

## I. INTRODUCTION

Suppose that a transmitter sends a message $a$ to a receiver through a communication channel with Shannon capacity $C_S$ by encoding message $a$. An $(n, W, \lambda_1)$ transmission code is a code which satisfies

$$\Pr[a \text{ is selected} | a \text{ is transmitted}] \geq 1 - \lambda_1 \tag{1}$$

for each message $a$, where each codeword has length $n$ and there are $W$ messages. The rate of the transmission code is defined as

$$R_1 \triangleq \log W/n.$$

Shannon proved that $\max R_1$ is equal to $C_S$ for any arbitrarily small $\lambda_1$. This model implicitly assumes the following.

1) A bijection from messages to codewords exists (deterministic encoding).
2) Also, the decoding regions of messages are disjoint and the receiver selects one message after receiving a noisy version of the transmitted codeword.

Ahlswede and Dueck [1] introduced a new model called identification codes via channels (ID codes). In this model

1) There are many codewords for each message and the transmitter chooses one of them randomly (probabilistic encoding).
2) The decoding regions of messages are not disjoint and the receiver chooses a list of messages after receiving a noisy version of the transmitted codeword.

An $(n, M, \lambda_1, \lambda_2)$ ID code is a code which satisfies (1) and

$$\Pr[b \text{ is selected} | a \text{ is transmitted}] \leq \lambda_2 \text{ for all } b \neq a \tag{2}$$

for each message $a$, where each codeword has length $n$ and there are $M$ messages. The probabilities are taken over the coin tosses of the transmitter (to choose a codeword) as well as over the noise of the channel. The rate of an ID code is defined as

$$R_2 \triangleq \log\log M/n$$

(which is a double $\log$ !!). It was proven that $\max R_2$ is equal to $C_S$ for any arbitrarily small $(\lambda_1, \lambda_2)$ [1], [5], [6].

In other words, any objects among $M = 2^{2^{nR_2}}$ objects (double exponentially many) can be identified in block length $n$ with arbitrarily small error probability if randomization can be used for the

Manuscript received July 14, 1998; revised March 1, 1999.

The authors are with the Department of Electrical and Electronic Engineering, Faculty of Engineering, Tokyo Institute of Technology, 2–12–1 O-okayama, Meguro-ku, Tokyo 152-8552, Japan.

Communicated by D. R. Stinson, Associate Editor for Complexity and Cryptography.

Publisher Item Identifier S 0018-9448(99)06071-X.