Paper from *European Trans. on Telecomm.*, Vol. 5, pp. 421-429, July-August 1994.

Some Applications of Source Coding in Cryptography

James L. Massey
Signal and Information Processing Laboratory
Swiss Federal Institute of Technology
ETH-Zentrum
CH-8092 Zürich

Abstract. It is shown that the techniques of source coding (or "data compression") can be usefully applied in cryptography. Five source coding schemes (Shannon-Fano coding, Huffman coding, Lynch-Davisson coding, Elias-Willems coding, and Lempel-Ziv coding) are reviewed and their characteristics delineated. It is then shown how the last three of these schemes, which are of the universal source coding type, can be advantageously used to create strongly-ideal ciphers, to perform "universal" homophonic substitution, to test random bit generators, and to strengthen running-key generators for stream ciphers.

Some Applications of Source Coding in Cryptography

James L. Massey
Signal and Information Processing Laboratory
Swiss Federal Institute of Technology
ETH-Zentrum
CH-8092 Zürich

1. Introduction

Whereas there has been a healthy influx of ideas from channel coding into cryptography, the concepts of source coding have had relatively little impact on cryptography. In this paper, we attempt to show that the techniques of source coding can readily be applied in cryptography.

In the next section, we present an introduction to the basic notions of source coding and point out the fundamental difference between "source-specific" and "universal" source-coding schemes. We then review five source coding schemes of considerable interest, three of which are of the universal kind. In the following section, we show how these universal source coding schemes in particular can be advantageously used to create strongly-ideal ciphers, to perform a new kind of "universal" homophonic substitution, to test random bit generators for statistical weaknesses, and to strengthen "weak" running-key generators for additive stream ciphers.

2. Principles of Source Coding

The goal of source coding is to represent the output of an information source with as few code digits per source letter as possible. In this paper, we consider only so-called *lossless* source coding (or "data compression") in which the source output can be reconstructed *exactly* from its compressed representation. We will also consider only binary encoding, both because this is the only case of real practical interest and because the generalization to arbitrary encoding alphabets is quite straightforward.

Let $U_1, U_2, U_3, \ldots$ denote the output sequence of a discrete information source. Such a source is said to be *stationary* if, for every positive integer L and every sequence $u_1, u_2, \ldots u_L$ of letters from the source alphabet, the probability $P(U_{i+1}, U_{i+2}, \ldots U_{i+L} = u_1, u_2, \ldots u_L)$ is the same for all $i \geq 0$. A stationary source is further said to be *ergodic* if the number of times that the sequence $u_1, u_2, \ldots u_L$ occurs within the source output sequence $U_1, U_2, \ldots, U_{N+L-1}$ of length N + L - 1, when divided by N, equals $P(U_1, U_2, \ldots U_L = u_1, u_2, \ldots u_L)$ with probability 1 as $N \rightarrow \infty$. In other words, a stationary source is ergodic when, with probability 1, it will emit an output sequence whose "time statistics" coincide with the ensemble statistics of the source. *Discrete stationary and ergodic sources* (DSES's) are general enough to model well any real information source so we will restrict our

attention to this class of information sources with no loss of essential generality.

All source-coding schemes of which we are aware can be modeled as the cascade of a *message parser* and a *message encoder* as shown in Fig. 1. The

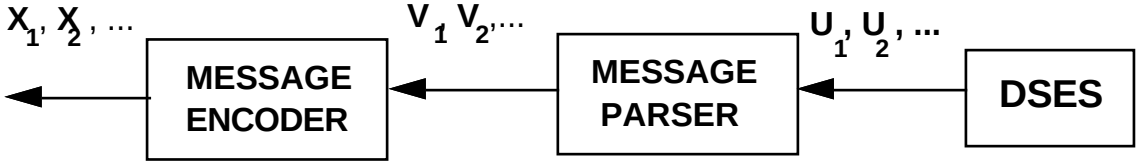


Fig. 1: A General Source Coding Scheme.

function of the message parser is to separate the source output sequence into phrases or messages. The simplest such message parser is the *L-block parser* for which each message consists of L source letters. For instance, the 3-block parser would parse the particular output sequence 001000100001000... of a binary DSES in the manner 001|000|100|001|000|... so that $\mathbf{V_1}$ = 001 is the first message, $\mathbf{V_2}$ = 000 is the second message, etc. A *variable-length parser*, however, might parse this same sequence as 0|0|1|00|01|000|010|00... so that $\mathbf{V_1} = \mathbf{V_2} = 0$, $\mathbf{V_3} = 1$, $\mathbf{V_4} = 00$, etc. Moreover, this same parser might parse a different source sequence into phrases of entirely different lengths. The function of the message encoder is to map each message, $\mathbf{V_i}$, into a binary codeword, $\mathbf{X_i}$, in such a manner that the message sequence can be recovered from the binary sequence formed by the concatenation of the codewords. A sufficient condition for such recovery is that this encoding be *prefix-free*, i.e. that no codeword for some value of $\mathbf{V_i}$ be the first part or "prefix" of the codeword for some other value of $\mathbf{V_i}$. This prefix-free condition is equivalent to the condition that a decoder be able to recognize the end of a codeword as soon as it sees the last digit of that codeword (so-called "instantaneous decoding"). It is well known (cf. [1, p. 49]) that weaker notions of recoverability of the message sequence from the codeword sequence do not lead to more efficient message encoding, so we will insist with no loss of real generality that the message encoding be prefix-free. However, the message encoder may have memory, i.e., the codeword for $\mathbf{V_i} = \mathbf{v}$ may depend on the values $\mathbf{V_1}, \mathbf{V_2}, ..., \mathbf{V_{i-1}}$, in which case the decoder will also have memory. On the other hand, every message parser is trivially reversible -- one simple removes the "markers" between phrases. Thus there is no limitation on the message parser of Fig. 1 (except the practical one that it be realizable) corresponding to the prefix-free limitation on the message encoder.

Let W_i denote the length of the binary codeword $\mathbf{X_i}$ in Fig. 1 and let Y_i denote the length in source letters of the corresponding message $\mathbf{V_i}$. The *rate* (in encoded bits per source letter) of the coding scheme is defined as

$$R = \lim_{n\to\infty} \left(\frac{W_1+W_2+ \ldots + W_n}{Y_1+Y_2+ \ldots + Y_n}\right). \qquad (1)$$

For each n, the ratio on the right of (1) is a random variable. However, if the message parser and message encoder can be realized by finite state machines as

we may assume with no loss of essential generality, then the ergodicity of the DSES ensures in general that this sequence of random variables tends to a constant with probability 1 as $n \to \infty$, so that the rate R is generally well defined by (1). The *information rate* (in bits per source letter) of a DSES is its per-letter entropy (in bits) defined as $H_\infty(U) = \lim_{n\to\infty} H_n(U)$ where

$$H_n(U) = \frac{1}{n} H(U_1 U_2 ... U_n). \qquad (2)$$

The *lossless source coding theorem* of information theory (cf. [1, p. 693]) states that source-coding schemes with $R < H_\infty(U)$ are impossible but that, for any $\varepsilon > 0$, there exist source-coding schemes with $R < H_\infty(U)+\varepsilon$. An optimum source-coding scheme is one whose rate R is (essentially) equal to $H_\infty(U)$.

If the source itself is binary (which is the usual case), then $\gamma = 1/R$ is called the *compression factor* of the source-coding (or "data-compression") scheme. An optimum scheme is one whose compression factor is essentially equal to $1/H_\infty(U)$. For any DSES, $H_\infty(U) \le H(U_1)$ with equality if and only if the source is *memoryless*, i.e. when its output is a sequence of independent and identically distributed (i.i.d.) random variables (cf. [1, p. 57]). For a binary memoryless source, $H(U_1) \le 1$ (bit) with equality if and only if $P(U_1 = 0) = P(U_1 = 1) = 1/2$, in which case the DSES is called the *binary symmetric source* (BSS). The output of the BSS is just a coin-tossing sequence for a fair coin. Because $H_\infty(U) \le 1$ for a binary DSES with equality if and only if the source is the BSS, it follows that *the BSS is the only binary source that cannot be compressed*, i.e., for which source coding with $\gamma > 1$ is impossible. But we may consider the binary sequence emitted by the source coding scheme in Fig. 1 itself to be the output of an information source. If this source coding scheme is optimum, then we cannot further compress this source so that it must be the BSS. Thus, one can view the *task of source coding* (or data compression) as the task of *reversibly transforming the source output to a coin-tossing sequence (or a good approximation thereto)*. This viewpoint is often useful in devising source-coding schemes and it is particularly useful in applying such schemes for cryptographic purposes.

3. Five Source-Coding Schemes

In this section, we review five source coding schemes of substantial theoretical and practical interest. Source-coding schemes can be classified as either *source specific* or *universal*. In a source-specific scheme, either the message parser or the message encoder (or both) of Fig. 1 requires explicit knowledge of the source statistics in order to perform its function. Such source-coding schemes tend to perform poorly for sources other than that for which they were designed. In a universal source-coding scheme, neither the message parser nor the message encoder requires knowledge of the source statistics. Such source-coding schemes tend to perform well (in an asymptotic sense) for all sources in some fairly large class. The first two schemes that we describe are of the source-specific type; the other three are universal. Our emphasis on universal schemes reflects our

conviction that such schemes are the most appropriate ones for cryptographic applications as the source statistics are seldom known with much precision in such applications. Our mention of two source-specific schemes is motivated by the fact that these two schemes best illustrate the fundamental principles of source coding.

3.1 Shannon-Fano Coding

Shannon-Fano coding [2, p. 53] is a technique for realizing the message encoder of Fig. 1 that explicitly aims to make the resulting sequence of codeword digits a good approximation to the output of the BSS. The Shannon-Fano algorithm is a "greedy" algorithm in the sense that it makes each subsequent codeword digit as nearly as equally likely to be a 0 or a 1 as possible, at the expense of possible severe biasing of later codeword digits. The algorithm is simple. One first makes a list of all possible messages in order of decreasing probability. Then one splits this list at the point where the two resulting sublists are as nearly equally probable as possible, assigning the first codeword digit as a 0 for messages in the first sublist and as 1 in the second sublist. One then repeats this splitting process on sublists to assign subsequent codeword digits to messages until all sublists contain a single message. The Shannon-Fano algorithm is illustrated in Fig. 2 for a set of five messages with probabilities 0.4, .15, .15, .15 and .15.

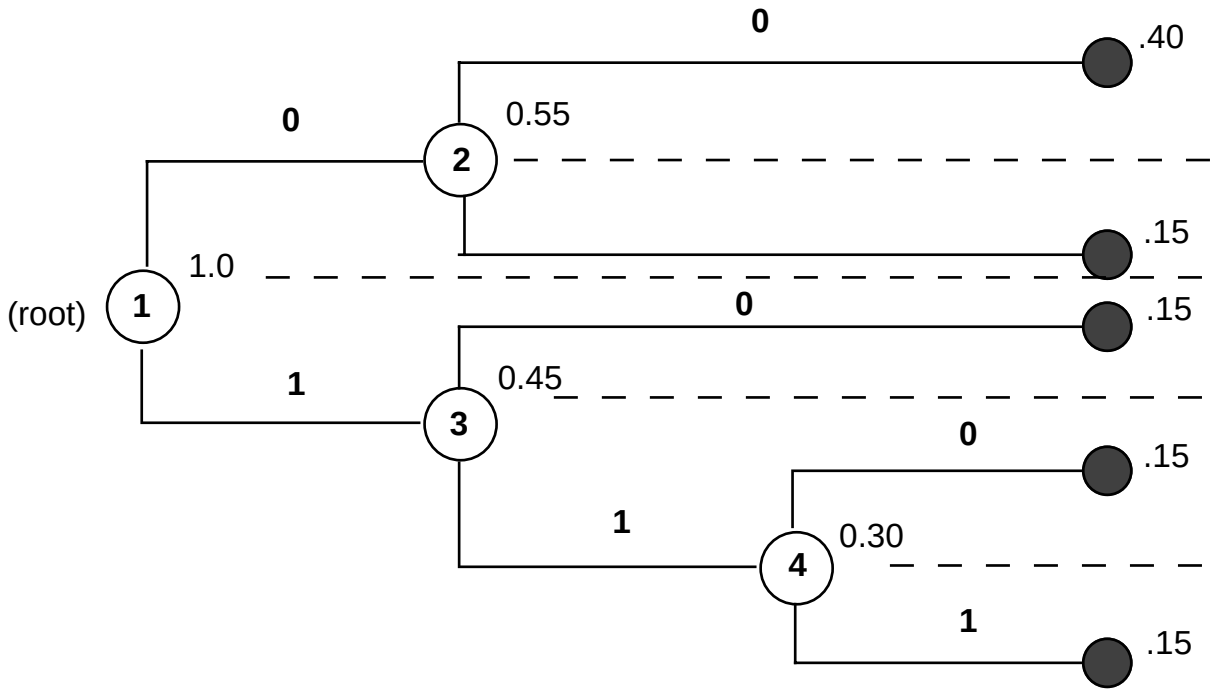


Fig. 2: An Example of Shannon-Fano Coding.

As Fig. 2 illustrates, the codewords in any prefix-free code correspond to the leaves of a binary tree, and conversely. Each node in Fig. 2 corresponds to a point of splitting in the Shannon-Fano algorithm and is labelled with the total

probability of all codewords that pass through this node. By the path-length lemma (cf. [3]), the average length of the codewords (i.e., the average depth of the leaves in the corresponding rooted tree with probabilities) is the sum of the node probabilities. Thus, the average length of the codewords in the Shannon-Fano code of Fig. 2 is E[W] = 2.30. We note from Fig. 2 that the first codeword digit is a 0 with probability 0.55, which is quite close to 1/2. However, given that the first digit is a 0, the second codeword digit has probability 0.4/(0.4 + .15) = 8/11 of being a 0, which is quite far from 1/2 and is the price of "greediness" in the choice of the first letter of the codewords.

3.2 Huffman Coding

The algorithm for optimum (i.e., minimum E[W]) prefix-free encoding of a message set was given by Huffman [4]. The trick is to be entirely "non-greedy" and to choose the last digits of codewords first. The algorithm is extremely simple. One assigns a last digit of 0 and 1, respectively, to the two least likely messages, then merges these two messages to a single message whose probability is the sum of those of the two merged messages. One then repeats this merging on the new message set until one has only a single message left. The Huffman algorithm is illustrated in Fig. 3 for the same message set used in Fig. 2. The average codeword length is seen by the path-length lemma to be E[W] = 2.20, which is slightly better than for the corresponding Shannon-Fano code. Note that the first codeword digit in the Huffman code has probability 0.40 of being a 0, which is not as close to 1/2 as for the Shannon-Fano code, but note also that there are no later badly biased digits in the Huffman code (in fact, for this example, all later codeword digits have probability exactly 1/2 of being a 0 in the Huffman code).

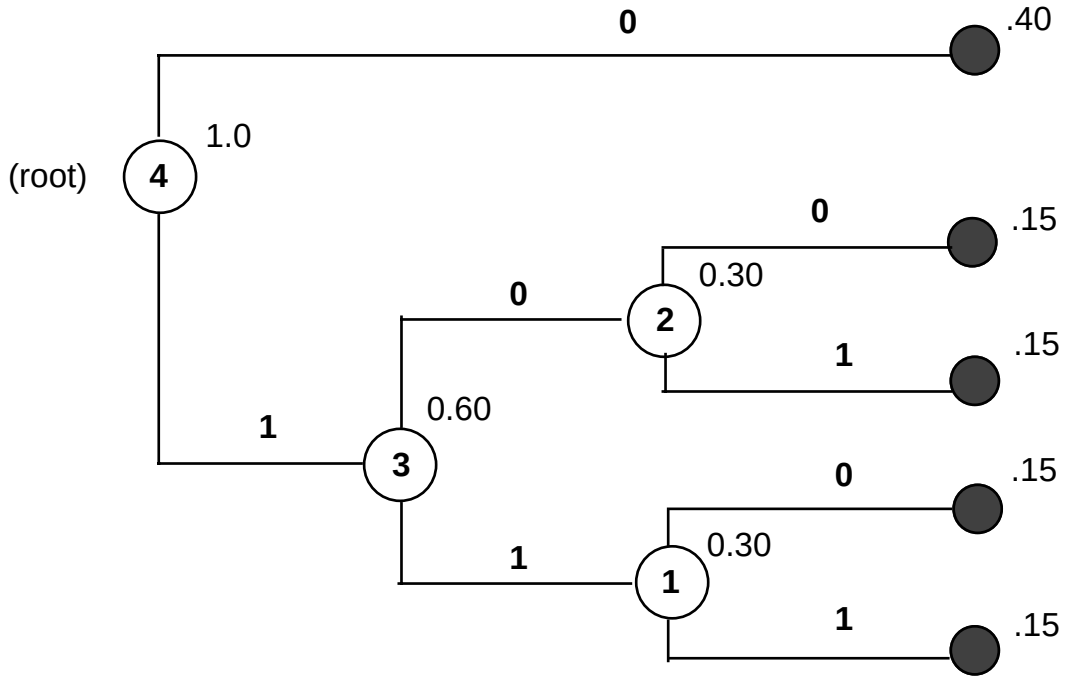


Fig. 3: An Example of Huffman Coding.

A standard argument in information theory (cf. [1, p. 58]) shows that if the Huffman code is used for the L-block message set, then the average length of the codeword $\mathbf{X_1}$ for $\mathbf{V_1} = [U_1, U_2, \ldots U_L]$ satisfies

$$H_L(U) \leq \frac{E[W_1]}{L} < H_L(U) + \frac{1}{L}. \tag{3}$$

Because the length Y_i of each message $\mathbf{V_i}$ is L for all i and because, for Huffman coding of each message, the length W_i of the i-th codeword $\mathbf{X_i}$ has the same probability distribution for all i so that $(W_1 + W_2 + \ldots + W_n)/n$ becomes equal to $E[W_1]$ with probability 1 as $n \to \infty$, it follows from (1), (2) and (3) that the rate R_L for Huffman coding of L-block messages satisfies

$$\lim_{L\to\infty} R_L = H_\infty(U). \tag{4}$$

Thus, any DSES can be optimally encoded in this manner by choosing L sufficiently large. The asymptotic optimality (4) can also be shown for the non-optimum (but often easier to implement) Shannon-Fano coding of the L-block message set.

The major drawback of Huffman coding (and also of Shannon-Fano coding) is that it requires knowledge of the message probabilities. [In practice, this is often done by "learning" these statistics from the past output of the source, which leads to so-called "adaptive Huffman coding".]

3.3 Lynch-Davisson Coding

The first, and still an interesting, universal source coding scheme was that independently given by Lynch [5] and Davisson [6], cf. also [2, pp. 139-142 and 164-167]. The Lynch-Davisson source coding scheme uses an L-block message parser. The message encoder first determines the Hamming weight (i.e., the number of 1's) W_H in the message $\mathbf{V_1} = [U_1, U_2, \ldots, U_L]$, then determines the index I of this message in an indexed list of all $\binom{L}{W_H}$ binary L-tuples of Hamming weight W_H. The codeword $\mathbf{X_1}$ is then the $\lceil \log(L + 1) \rceil$ bit binary code for W_H followed by the $\lceil \log \binom{L}{W_H} \rceil$ bit binary code for I, where here and hereafter all logarithms are to the base 2. Because the length of the code for W_H does not depend on the particular message $\mathbf{V_1}$, the decoder can determine W_H from this code to determine where the codeword will end, so this encoding of the message $\mathbf{V_1}$ is indeed prefix-free.

For practical implementation, one needs a simple algorithm to determine the index I from $\mathbf{V_1}$, and vice-versa. The heart of Lynch-Davisson coding is the simple algorithms for this enumeration and its inverse. The index of **v** in the list

of all L-tuples with Hamming weight w is taken simply as its lexicographical order in this list, where 0 precedes 1 and where the index of the first entry is zero. If the first 1 in **v** occurs in position n, then **v** is preceded lexicographically by all $\binom{L-n}{w}$ L-tuples of Hamming weight w that begin with n 0's, hence the index I of v is $\binom{L-n}{w}$ plus the index in the list of L-tuples of Hamming weight w-1 of the word obtained by setting the n-th bit of **v** to 0. Thus, if the 1's in **v** occur at positions n_i for i = 1, 2, ..., w, where $n_i < n_{i+1}$, then the index of **v** is just

$$I = \sum_{i=1}^{w} \binom{L-n_i}{w-i+1}$$

and thus is easily computed from a stored table of binomial coefficients. The inverse mapping is easily accomplished recursively from the fact that n_1 is the smallest integer n such that $I > \binom{L-n}{w}$, i.e., such that **v** is preceded lexicographically by all the L-tuples of Hamming weight w that begin with n leading 0's.

Suppose that the information source is a binary memoryless source (BMS) with $P(U_1 = 1) = p$. The information rate is $H_\infty(U) = H(U_1) = h(p)$ where $h(p) = - p \log p - (1-p) \log (1-p)$ is the binary entropy function. For large L, the law of large numbers ensures that W_H / L will be close to p with high probability. Thus, the length W of the Lynch-Davisson codeword will be close to $\lceil \log (L+1) \rceil + \log \binom{L}{pL}$ with high probability. But $\log \binom{L}{pL} / L \approx h(p)$ so that the rate R_L of the Lynch-Davisson code will approach $H_\infty(U)$ as $L \rightarrow \infty$ for any BMS. One says that the Lynch-Davisson source-coding *scheme is universally asymptotically optimum (or simply "universal") for the class of all binary memoryless sources.* As one might expect, it also works well for DSES's with weak memory [i.e., $H_\infty(U) \approx H(U_1)$], but can be very inefficient for such sources with strong memory [i.e., $H_\infty(U) << H(U_1)$].

3.4 Elias-Willems Coding

Before describing the next source-coding scheme, we first digress to consider Elias' two prefix-free coding schemes [7] for the positive integers $\mathbb{Z}^+ = \{1, 2, 3,...\}$. Consider the natural binary coding B(n) for $n \in \mathbb{Z}^+$, i.e., B(1) = 1, B(2) = 10, etc. We note that this natural binary code is not a prefix-free code for $\mathbb{Z}^+$ [in fact B(1) = 1 is a prefix of every other codeword] and that the length L(n) of B(n) is $\lfloor \log n \rfloor + 1$. Elias' first coding scheme for $\mathbb{Z}^+$ encodes n as $B_1(n)$ where $B_1(n)$ consists of L(n) - 1 0's followed by B(n). For instance, because $L(13) = \lfloor \log 13 \rfloor + 1 = 4$, one obtains $B_1(n) = 0001101$. The length of $B_1(n)$ is $L_1(n) = 2\, L(n) - 1 = 2 \lfloor \log n \rfloor + 1$, about twice that of B(n). However, the encoding $B_1(n)$ is prefix-free because the number L(n) - 1 of leading 0's in $B_1(n)$ determines the length of the codeword, i.e., where the codeword will end. Elias's second prefix-free coding scheme for $\mathbb{Z}^+$ builds on the first. The codeword $B_2(n)$ is $B_1(L(n))$ [i.e., the first coding applied to the length of n in the natural binary code] followed by B(n) with its now "useless" leading 1 removed. [Elias was interested in asymptotic results in [7] and did not bother with removing this unneeded digit, but this refinement is important in practical applications.] For instance, because L(13) = 4, $B_1(4) = 00100$, and B(13) =

1101, one obtains $B_2(13) = 00100101$. The length of $B_2(n)$ is easily computed to be

$$L_2(n) = \lfloor \log n \rfloor + 2 \lfloor \log (\log n + 1) \rfloor + 1 \qquad (5)$$

so that $L_2(n)/L(n) \approx 1$ for large n.

A direct check shows that $L_1(n) = L_2(n)$ for $n = 1$ and $4 \le n \le 7$ and $16 \le n \le 31$, that $L_1(n) = L_2(n) - 1$ for $2 \le n \le 3$ and $8 \le n \le 15$, but that $L_1(n) > L_2(n)$ for all $n \ge 32$. Thus, the "asymptotic superiority" of $B_2(n)$ over $B_1(n)$ begins to prevail already for rather small n.

For any positive-integer-valued random variable J with large entropy such that

$$P(J = n+1) \le P(J = n), \quad n \in \mathbb{Z}^+, \qquad (6)$$

$B_2(n)$ is an essentially optimum prefix-free encoding of J in the sense that the expected codeword length $E[L_2(J)]$ will be close to its minimum possible value, which is the entropy H(J) of J. In other words, (6) together with $H(J) >> 1$ bit ensures that

$$\frac{E[L_2(J)]}{H(J)} \approx 1. \qquad (7)$$

This follows from the fact that (6) implies that $P(J = n) \le 1/n$ and hence that $-\log P(J = n) \ge \log n$, which further implies that $H(J) \ge E[\log J]$; but (5) gives $E[\log J]/E[L_2(J)] \approx 1$ when $H(J) >> 1$ bit.

In the Elias-Willems source-coding scheme, as we shall call the scheme in this section which is based on ideas introduced independently by Elias [8] and Willems [9], the message parser is again the L-block parser. The message encoder first transforms the message $\mathbf{V_i}$ to a positive integer J_i that indicates how long it has been since that message was last seen. For instance, if $\mathbf{V_i} = \mathbf{v}$ but $\mathbf{V_{i-1}} \ne \mathbf{v}$, $\mathbf{V_{i-2}} \ne \mathbf{v}$ and $\mathbf{V_{i-3}} = \mathbf{v}$, then $J_i = 3$. [This scheme clearly requires an initialization, say by arbitrarily assigning times of last occurrence from $-(2^L-1)$ to 0 to the 2^L possible L-block messages. We ignore this unimportant practical detail and assume that the coding system is already in its "steady state" when encoding first begins.] The message $\mathbf{V_i}$ is then encoded as $\mathbf{X_i} = B_2(J_i)$, i.e., as Elias' second coding applied to the time since the most recent previous occurrence of $\mathbf{V_i}$. [Elias [8] describes a scheme similar to that described here but again he does not bother to remove the "useless" 1 in his second coding scheme; this removal is usually important in practical applications where L cannot be too large.]

The ergodicity of a DSES implies that the relative frequency of a particular message $\mathbf{v}$ in the message sequence $\mathbf{V_1}, \mathbf{V_2}, \mathbf{V_3}, \ldots$ will equal $P(\mathbf{V_1} = \mathbf{v})$ with probability 1. Thus, the time average value of J_i when $\mathbf{V_i} = \mathbf{v}$ must be exactly $1/P(\mathbf{V_1} = \mathbf{v})$. Again by ergodicity, this time average must equal the corresponding

conditional expectation so that

$$E[\, J_1 \mid \mathbf{V_1} = \mathbf{v}] = 1 / P(\mathbf{V_1} = \mathbf{v}). \tag{8}$$

Equation (8) suggests that small values of J_1 arise from messages with large probability and hence should be more likely than large values of J_1. Thus condition (6) should be fulfilled at least roughly, which is the main motivation for using $B_2(n)$ to code J_1. But $H(\mathbf{V_1}) = LH_L(U)$ and $E[W_1] = E[L_2(J_1)]$. Thus, when $H(\mathbf{V_1}) >> 1$ bit (which will be the case for sufficiently large L for any DSES with $H_\infty(U) \neq 0$), the rough satisfaction of (6) should ensure that (7), which now becomes

$$\frac{E[W_1]}{L} \approx H_L(U), \tag{9}$$

should be satisfied when $H(\mathbf{V_1}) >> 1$ bit. This can be rigorously shown to be true by starting from (8) and (5), then overbounding $E[W_1 \mid \mathbf{V} = \mathbf{v}]$ by making repeated use of Jensen's inequality for the expectation of concave functions of random variables, and finally averaging over **v** with a further use of Jensen's inequality. But (9) implies that the rate R_L of this Elias-Willems source coding scheme satisfies

$$\lim_{L \to \infty} R_L = H_\infty(U) \tag{10}$$

for any DSES. Thus, *the Elias-Willems source-coding scheme is universal for the class of all discrete stationary and ergodic sources.*

3.5 Lempel-Ziv Coding

Unlike the two previous universal source-coding schemes, the Lempel-Ziv scheme uses variable-length message parsing; indeed this parsing is its most distinctive feature. There are rather many versions of the Lempel-Ziv scheme, all of which are based on the ideas originally proposed in [10]. We will consider the version described by Welch [11], which seems to be the one most often implemented, and we will refer to this version as the *LZ-W source-coding scheme*.

The key idea in any Lempel-Ziv source-coding scheme is to parse the source sequence according to its *innovations*, i.e., according to the subsequences or "strings" that appear for the first time within the source sequence. In the LZ-W version, one parses a binary source by assuming that the length-one strings 0 and 1 are the only previously encountered strings. Let $L_1 = (0,1)$ denote this initial list. The parsing rule is then as follows. For each i, i = 1, 2, ..., mark the end of the i-th phrase at the point where including the next digit would give a string not in the list L_i of previously encountered strings, then place this string with the next digit appended at the end of the list L_i to form the list L_{i+1}. Applying this parsing rule to the sequence 001000100001000... gives

0|0|1|00|01|000|010|00..

as we now explain. The initial string 0 is in $L_1 = (0,1)$, but the string 00 is not. Thus, we place a marker after the initial 0 and form the list $L_2 = (0,1,00)$. Looking forward from this first marker, we first see 0, which is in L_2, then we see 01, which is not. Thus we place a marker after this second 0 and form the list $L_3 = (0,1,00,01)$, etc.

The messages $\mathbf{V_1}, \mathbf{V_2}, \mathbf{V_3}, \ldots$ of the LZ-W scheme are the phrases of the parsed source sequence. Note that the list L_i contains exactly i+1 strings. In the LZ-W scheme, the message $\mathbf{V_i}$ is encoded as the $W_i = \lceil \log (i+1) \rceil$ bit binary code for its index in the list L_i. [Note that, for i > 1, the last string in the list L_i is placed there only after the parsing of $\mathbf{V_{i-1}}$, which requires examination of the first digit of $\mathbf{V_i}$. Thus, for i > 1, the decoding of the codeword $\mathbf{X_i}$ to the message $\mathbf{V_i}$, when $\mathbf{X_i}$ is the codeword pointing to the last entry in L_i, cannot be performed by table look-up as the decoder will then have formed only the list L_{i-1}. But the last entry in L_i is always a string having $\mathbf{V_{i-1}}$ as a prefix. Thus, when i > 1 and $\mathbf{X_i}$ points to this last string in L_i, the first digit of $\mathbf{V_i}$ must be the same as the first digit of $\mathbf{V_{i-1}}$ and hence the decoder can "prematurely" form the list L_i that it needs to decode $\mathbf{X_i}$.]

Because the length W_i of the i-th codeword $\mathbf{X_i}$ does not depend on the source sequence, the LZ-W coding is prefix-free; moreover, the lengths of the first n codewords sum to

$$\sum_{i=1}^{n} W_i = \sum_{i=1}^{n} \lceil \log (i+1) \rceil .$$

The corresponding sum of message lengths, however, depends strongly on the statistics of the DSES encoded. Lempel and Ziv [10] have shown (by an argument that applies also to the LZ-W version) that, for any DSES, the code rate R of (1) satisfies

$$R = H_\infty(U),$$

i.e., that *the Lempel-Ziv source-coding scheme is universal for the class of all discrete stationary and ergodic sources.*

As we have described the LZ-W scheme, the list of previously encountered strings grows without limit. In practice, one generally places a limit (say 2^m) on the size of this list so that m becomes a system parameter. When the maximum list size is reached, one either restarts the algorithm or reverts to a new parsing in which successive parsing marks are placed after the longest string still in the list so that encoding and decoding can continue with this largest list. Lempel-Ziv source coding, and in particular the LZ-W version, has proved to be a very popular data-compression scheme in practice, as much because of the ease with

which it can be implemented as because of its universality.

4. Cryptographic Applications

In this section, we describe several potential applications of source coding in cryptography. In particular, we show the cryptographic utility of the three universal source-coding schemes described in the previous section. We wish to stress here that the applications that we describe are intended to be *illustrative*, not exhaustive. Other possible applications of source coding in cryptography will undoubtedly occur to the thoughtful reader. We consider, in particular only secret-key (or "symmetric") ciphers. We will also always assume that the plaintext source is a binary information source and that the ciphertext digits are also binary.

4.1 Creating Strongly-Ideal Ciphers

By a *non-expanding cipher*, we mean a cipher for which there is an increasing sequence of positive integers $n_1, n_2, n_3, \ldots$ such that the first n_i digits $Y_1, Y_2, \ldots Y_{n_i}$ of the ciphertext together with the secret key uniquely determine the first n_i digits $X_1, X_2, \ldots, X_{n_i}$ of the plaintext for $i = 1, 2, 3, \ldots$. *Additive stream ciphers* in which $Y_i = X_i \oplus Z'_i$, where $Z'_1, Z'_2, Z'_3, \ldots$ is the *running-key* generated from the secret key **Z** are non-expanding; one can simply choose $n_i = i$. *Block ciphers*, in which the plaintext and ciphertext blocks both have the same length N, are also non-expanding; one can choose $n_i = iN$. Our interest in non-expanding ciphers stems from the following fact.

Random-in/Random-out Property of Non-Expanding Ciphers : For every choice **z** of the secret key **Z**, the cascade of a binary symmetric source (BSS) and a non-expanding cipher is another BSS. Moreover, for any probability distribution for the secret key, the cascade of a BSS with a non-expanding cipher yields a ciphertext sequence $Y_1, Y_2, Y_3, \ldots$ that is statistically independent of the secret key **Z.**

To prove this fact, we must show, for any choice **z** of the secret key **Z** and any positive integer m, that the ciphertext sequence $Y_1, Y_2, \ldots Y_m$ of length m is equally likely to be equal to any of the 2^m binary sequences of length m. Let n, with $n \geq m$, be such that $Y_1, Y_2, \ldots Y_n$ uniquely determine $X_1, X_2, \ldots X_n$. Because all 2^n possible values of the BSS output sequence $X_1, X_2, \ldots X_n$ of length n are equally likely, it follows that all 2^n possible values of $Y_1, Y_2, \ldots Y_n$ must also be equally likely. Thus, $Y_1, Y_2, \ldots Y_m$ must also be a BSS output sequence of length m so that all 2^m values of this sequence are also equally likely, as was to be shown. Our argument shows further that, for every m, $P(Y_1, Y_2, \ldots Y_m = y_1, y_2, \ldots y_m \mid \mathbf{Z} = \mathbf{z}) = 2^{-m}$, independent of **z**, and hence that the output sequence $Y_1, Y_2, Y_3, \ldots$ is indeed statistically independent of **Z**, regardless of the probability distribution for **Z.**

Shannon [12] has defined a cipher to be *strongly ideal* if the secret key **Z** is

statistically independent of the ciphertext sequence $Y_1, Y_2, Y_3, \ldots$. In a *ciphertext-only* attack on a strongly ideal cipher, an attacker can obtain no information about the secret key **Z** in effect, no matter how much ciphertext he or she examines, i.e., the security of the cipher does not diminish with the quantity of plaintext enciphered before changing of the secret key. The random-in/random-out property above implies that *every non-expanding cipher is strongly ideal when the plaintext source is a BSS.*

Because a "perfect" source-coding scheme converts the information source into a BSS, as we discussed in Section 2, it follows that *applying perfect source coding to an information source before encryption with a non-expanding cipher creates a strongly ideal cipher.* No practical source-coding scheme for a realistic information source, however, is perfect; there is always some residual *redundancy* $\rho = 1 - H_\infty(X)$ in the compressed sequence $X_1, X_2, X_3, \ldots$ used as plaintext. Such compression is still very useful since it increases the unicity distance of the cipher (i.e., the least amount of ciphertext for which, with sufficient effort, the attacker in a ciphertext-only attack can find the secret key). According to Shannon's celebrated formula [12], the unicity distance can generally be well estimated as

$$n_u = \frac{H(Z)}{\rho} \ . \qquad (11)$$

Thus, even such "imperfect" source coding can greatly strengthen a cipher by reducing the redundancy ρ and thus increasing its unicity distance.

The above considerations show the value of compressing any real information source by a suitably universal source-coding scheme, such as Elias-Willems coding or Lempel-Ziv codings, that will yield low redundancy ρ. A word of caution, however, is in order. Both of these universal source-coding schemes pass through an "initialization phase" before they settle down to efficient coding. In Elias-Willems coding, this initial phase is needed to remove the effect of the arbitrary choice of most recent times of occurrence of the possible messages that is required when coding first begins. In Lempel-Ziv coding, this initial phase is needed to produce many typical long source phrases in the list of previously encountered phrases. Thus, for both of these universal source-coding systems, it is wise, whenever possible, to perform "plaintext stuffing" (i.e., the insertion of truly random bits at pre-arranged positions in the plaintext sequence) on the first suitably-long portion of the plaintext to ensure redundancy reduction up to the point where the universal source-coding scheme can be expected to reach essentially its asymptotic efficiency in redundancy removal.

Of course, removing redundancy from the plaintext is an old cryptographic trick for increasing the security of a cipher. Shannon [12] explicitly recommends this practice. In the earlier days of hand enciphering, redundancy was often removed by deleting "unneeded" letters from the plaintext -- ths s n xmpl. Universal source-coding, however, now offers quite practical ways to reduce the

redundancy of a plaintext source in a scientific manner, even for very high speed encipherment.

4.2 Universal Homophonic Coding

The general form of *homophonic coding* (or "multiple substitution") is indicated in Fig. 4. The output of a BSS is the *randomizer* that is used to map the output of the actual information source randomly into the plaintext sequence in such a way that the actual source sequence can be recovered from the plaintext sequence without knowledge of the randomizer. In this way, many particular plaintext sequences become possible substitutes (or "homophones") for a particular source sequence, the choice being determined by the particular randomizer sequence. Perfect homophonic coding makes the resulting plaintext sequence a BSS output sequence, with all the advantages described in Section 4.1. For recent treatments of such perfect homophonic coding, see [13] and [14]. These perfect homophonic coding schemes, however, have a major practical drawback in that they require specific and exact knowledge of the statistics of the actual information source.

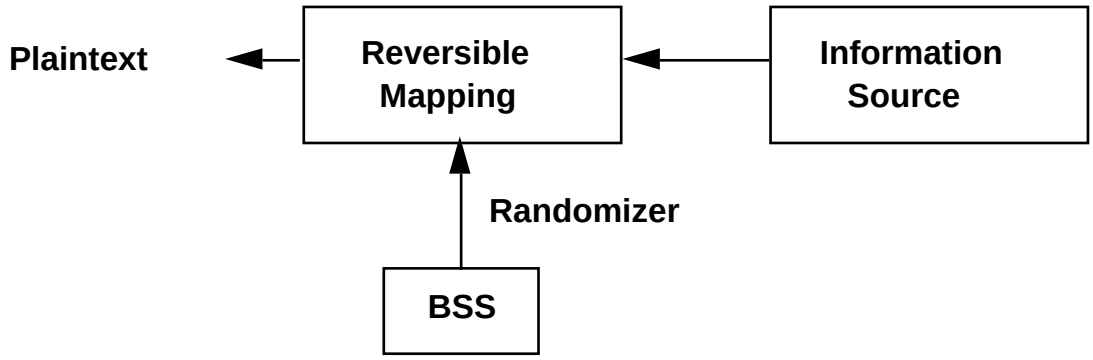


Fig. 4: General Homophonic Coding.

To avoid having to know the source statistics in order to perform homophonic coding, we propose in Fig. 5 a kind of *universal homophonic coding*. By the (m,n) multiplexer shown in Fig. 5, we mean a device that first outputs m randomizer digits from the BSS, then n digits from the actual binary information source, then the next m randomizer digits from the BSS, then the next n digits from the information source, etc. The homophonic coder output is then obtained by processing the multiplexer output with a suitably universal source-encoding scheme. The actual information source sequence can be recovered from the homophonic coder output without knowledge of the randomizer simply by passing this output through the decoder for the source-coding scheme and then discarding the randomizer digits.

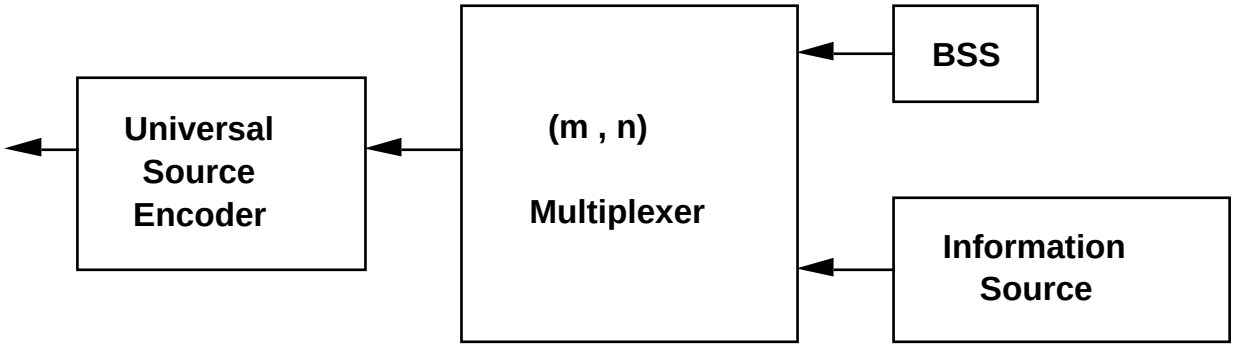


Fig. 5: Universal Homophonic Coding.

Except when the actual information source, which we assume to be a discrete stationary and ergodic source (DSES), is also a BSS (in which case there would be no need for homophonic coding), the output sequence of the multiplexer is not that of a true DSES as the multiplexing introduces non-stationarity. However, the sequence $\mathbf{V_1}, \mathbf{V_2}, \mathbf{V_3}, \ldots$ of L-block messages from this source do indeed form the output of a DSES (whose output alphabet is the set of binary L-tuples) whenever L is a multiple of m + n. This message source is easily seen to have an information rate

$$H_\infty(V) = m + nH_\infty(U) \tag{12}$$

where $H_\infty(U)$ is the information rate of the actual binary information source. This implies that the Elias-Willems scheme will efficiently compress this message source, and that the simple Lynch-Davisson code will also work reasonably well, particularly when $m \geq n$, since then this message source will have rather weak memory as $H_\infty(\mathbf{V}) \geq m$ bits and $H(\mathbf{V_1}) \leq m + n$ bits. Either of these universal source-coding schemes when used with an L-block parser for which L is a multiple of m + n will make the universal homophonic coding scheme of Fig. 5 very strong indeed.

It is also possible to view the homophonic coding scheme of Fig. 5 as a generalized form of "plaintext stuffing". We mention this fact only to make it evident that the "initialization problem" encountered in Section 4.1 hardly arises for the universal homophonic coding scheme of Fig. 5.

4.3 Statistical Testing of Random Bit Generators

By a *random bit generator* (RBG), we mean a physical device intended to be a realization of a BSS. Such RBG's are of much utility in cryptography, particularly for the generation of secret keys, but also for such purposes as providing the bits for plaintext stuffing or providing the randomizer for homophonic coding. A perennial problem in cryptography is how to make statistical tests on the output of an RBG to ensure that it is a sufficiently faithful realization of a BSS. Maurer [15] has recently given a thorough treatment of how universal source coding provides a natural solution to this testing problem so we

will content ourselves here with outlining Maurer's arguments and theory.

The essence of Maurer's approach is the observation that defective RBG's are generally defective in such a way that they can be well modelled as a DSES that is not the BSS. For instance, an RBG with an internal thermal noise generator and threshold detector can be expected to generate independent digits with some probability p of being a 1; such an RBG is thus well modelled as a binary memoryless source. If the output of a possibly defective RBG is processed by a suitably universal source-coding scheme, the rate R of this encoding will be a good approximation to the information rate of the RBG. Thus, an R significantly less than 1 will signal a defective RBG. The beauty of this approach lies in the fact that 1 - R provides an objective measure of the degree of departure of the RBG from the BSS that does not depend on more or less arbitrary "confidence levels" or "significance levels". From (1), one sees that the determination of R requires knowledge only of the lengths of codewords and not of the codewords themselves. The happy consequence is that one can shortcut the source-coding algorithm to deliver only lengths of codewords when using this algorithm for testing of RBG's.

In [15], Maurer proposes using essentially Elias-Willems coding to test RBG's in the manner just described, except that he replaces the exact codeword length $L_2(J_i)$ as given by (5) with the simple but sufficiently accurate approximation $\log(J_i)$, which has the further virtue that its statistics are readily computable for the true BSS. Maurer's paper provides a wealth of information to assist one in a practical implementation of this test.

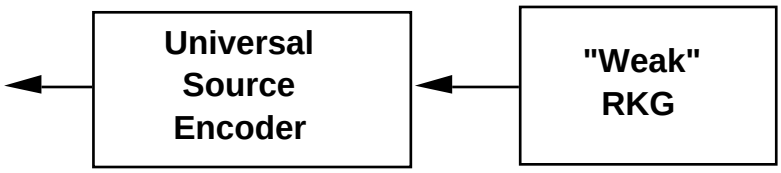
If it is known that the possibly defective RBG is well modelled by a BMS, then it would be preferable to base the statistical test for such an RBG on Lynch-Davisson coding rather than Elias-Willems coding, as the convergence of R_L to R for the former and simpler scheme is essentially optimally fast among source-coding schemes with L-block parsing that are universal for the class of all BMS's. The use of the more-widely-universal Elias-Willems coding will, however, rescue the test should the RBG, contrary to assumption, not in fact be well modelled by a BMS.

4.4. Strengthening Running-Key Generators

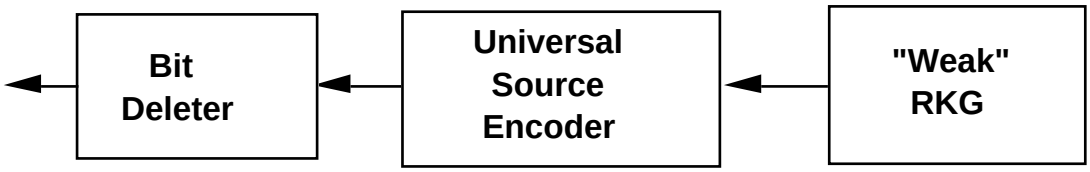
In this section, we propose an application of universal source-coding schemes in cryptography that is considerably more speculative than the three previously described applications. We propose using such schemes to strengthen "weak" running-key generators. By a running-key generator (RKG), we mean a finite-state machine which, when initialized in some manner by the secret key **Z**, produces the running key $Z'_1, Z'_2, Z'_3, \ldots$ for an additive stream cipher. The goal of RKG design is to produce a running key that is difficult to predict reliably from its past history, when one does not know the secret key Z. For instance,, a linear feedback shift-register (LFSR) of length L, whose initial state is the secret key, is a

"weak" RKG as it is easy to predict the future of the resulting running key without error from any 2L consecutive digits of this sequence [16].

In a certain sense, the "ideal" RKG would be equivalent to a BSS to an attacker ignorant of the secret key. Given any number of past output bits from a BSS, the attacker can predict the next bit with an error probability no better (and no worse) than 1/2. Because good source-coding schemes for a discrete stationary and ergodic source (DSES) convert such a source to a good approximation to a BSS, it might seem natural to try to strengthen a "weak" (i.e., easily predictable) RKG by merely applying such a data compression scheme to its output as shown in Fig. 6(a). This approach, however, is naive for two reasons. First, by simply applying the decoder of the source-coding scheme to the output of the system of Fig. 6(a), an attacker can recover the running-key of the "weak" RKG so that there has in fact been no real strengthening of its unpredictability (although errors in such prediction might propagate more because of the subsequent source coding). Second, and more fundamentally if less important, the "weak" RKG cannot be well modelled by a DSES. The reason is that the entropy of any portion of the running key, no matter how long, cannot exceed the entropy of the secret key **Z**, which is the only source of randomness in the running key.



(a) Naive Version.



(b) Proposed Version.

Fig. 6: Strengthening a "Weak" RKG with Source Coding.

To convert the naive system of Fig. 6(a) into a useful "security amplifier" for a "weak" RKG requires first that we destroy the inherent invertibility of the source-coding scheme. This invertibility is essential in true source-coding schemes, but it is not at all necessary when the source coding scheme is being used only to increase the randomness of some sequence. A simple and effective way to do this is by *bit deletion*, i.e., by deleting the digits in prearranged positions in the output of the source encoder. The strengthened RKG obtained in this manner is shown in Fig. 6(b). In general, one will want to delete more bits during the initialization phase of the source encoding than during the "steady state" and one will also want to choose the bits to be deleted with some care to maximize

the resulting confusion for an attacker who does not know the deletion pattern. For instance, if Elias-Willems coding is used, one will probably want to delete all of the leading 0's and the first 1 of each codeword, as this will make it extremely difficult to determine the boundaries between codewords. If Lempel-Ziv coding is used, one will probably want to delete the most significant bits of each codeword as this will make it very difficult to find the location in the list of previously encountered strings to which this codeword is pointing. If Lynch-Davisson coding is used, one will probably want to delete the "Hamming weight" portion of the codeword as the remainder of the codeword will then appear very random indeed.

Of course any RKG, whether "weak" or not, cannot be well modelled as a DSES for the reason mentioned above, which casts some suspicion on the use of source coding to strengthen RKG's in the manner shown in Fig. 6(b). Nonetheless, we anticipate that such security amplifiers would actually work quite well for the reason that even a "cryptographically weak" RKG should produce an output sequence that is difficult to distinguish from a DSES output sequence. Only if a universal source coding scheme with a large parameter L were allowed to operate for a very long time would one expect this source-coding scheme to compress such an information source near to its true information rate of 0 bits per source symbol. Even "cryptographically weak" RKG's should be confusing enough to a universal source-coding scheme that such a scheme will requires a very long time to learn the real nature of such a source.

5. Concluding Remarks

It should be apparent from the above that we have only skimmed the surface of the possible applications of source coding in cryptography. We will consider it to be a very satisfactory outcome of this paper if it should stimulate the reader to devise creative new cryptographic applications of source coding.

References

[1] R. G. Gallager, *Information Theory and Reliable Communications*. New York: Wiley, 1968.

[2] T. J. Lynch, *Data Compression: Techniques and Applications*. Belmont, CA: Lifetime Learning (Wadsworth), 1985.

[3] J. L. Massey, "An Information-Theoretic Approach to Algorithms," pp. 3-20 in *The Impact of Processing Techniques in Communications,* (Ed. J. K. Skwirzynski) NATO Advanced Study Institutes Series E91. Dordrecht, The Netherlands: Nijhoff, 1985, .

[4] D. A. Huffman, "A Method for the Construction of Minimum Redundancy Codes," *Proc. IRE*, Vol. 40, pp. 1098-1101, Sept. 1952.

[5] T. J. Lynch, "Sequence Time Coding for Data Compression," *Proc. IEEE*, Vol. 54, pp. 1490-1491, Oct. 1966.

[6] L. D. Davisson, "Comments on 'Sequence Time Coding for Data Compression'", *Proc. IEEE*, Vol. 54, p. 2010, Dec. 1966.

[7] P. Elias, "Universal Codeword Sets and Representations of the Integers," *IEEE Trans. Inform. Th.*, Vol. IT-21, pp. 194-203, March 1975.

[8] P. Elias, "Interval and Recency Rank Coding: Two On-Line Adaptive Variable-Length Schemes," *IEEE Trans. Inform. Th.*, Vol. IT-33, pp. 3-10, Jan. 1987.

[9] F. M. J. Willems, "Universal Data Compression and Repetition Times," *IEEE Trans. Inform. Th.*, Vol. IT-35, pp. 54-58, Jan. 1989.

[10] J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression," *IEEE Trans. Inform. Th.*, Vol. IT-23, pp. 337-343, May 1977.

[11] T. A. Welch, "A Technique for High Performance Data Compression," *IEEE Computer*, Vol. 17, pp. 8-19, June 1984.

[12] C. E. Shannon, "Communication Theory of Secrecy Systems", *Bell Sys. Tech. J.*, vol. 28, pp. 656-7l5, Oct. l949.

[13] C. G. Günther, "A Universal Algorithm for Homophonic Coding", pp. 405-414 in *Advances in Cryptology - Eurocrypt '88* (Ed. C. G. Günther, Lect. Notes in Comp. Sci. No. 330. Heidelberg and New York: Springer l988.

[14] H. K. Jendal, Y. J. B. Kuhn, and J. L. Massey, "An Information-Theoretic Approach to Homophonic Substitution," pp. 382-394 in *Advances in Cryptology-Eurocrypt '89* (Eds. J.-J. Quisquater and J. Vandewalle), Lect. Notes in Comp. Sci., No. 434. Heidelberg and New York: Springer, 1990.

[15] U. M. Maurer, "A Universal Statistical Test for Random Bit Generators," *J. Cryptology*, Vol. 5, No. 2, pp. 89-105, 1992.

[16] J. L. Massey, "Shift-Register Synthesis and BCH Decoding," *IEEE Trans. on Info. Th.*, Vol. IT-15, pp. 122-127, Jan. 1969.