A New Novel Based Approach to Enhance Security in Arithmetic Coding using Random Probabilities

N. Karthikeyan[1]

[1]Assistant Professor
Department of Mathematics, Kongunadu College of Engineering and Technology
Tiruchirappalli - 621 215, India
karthi.lect@gmail.com

Abstract: ***The current network scenario demands secure and fast communication of information. This requires both compression and encryption. Arithmetic coding provides compression as well as data security. In fixed model arithmetic coding security is improved by assigning symbol frequencies arbitrarily rather than matching the frequency of occurrence in the source input and by means of random symbol ordering. In this paper security is enhanced by randomly assigning probabilities to frequency of occurrence using pseudo random numbers and finding the product between frequency of occurrence of the character and randomly selected probabilities and by not altering the frequency of occurrence and symbol ordering. The associated frequencies of the symbols do not match with the actual frequencies of the symbols.***

Keywords: Data security, Arithmetic coding, Random probability, seed value generation

1. Introduction

In today's e-transaction driven world, the present network scenario demands secure and effective e-transaction or exchange of information. This can be accomplished by compression and encryption. Coding may be described by which an encoded string is produced by a model from an input string of symbols. Encryption is indeed a secure coding technique in which plain text is transformed into cipher by the use of a model in conjunction with some secret information (key). Data compression is also a coding technique, whose purpose is to reduce both the space requirements for data storage and the time for data transmission [1]. Information theory an entropy encoding is a loss less data compression scheme that is independent of the specific characteristics of the medium. One of the main types of entropy coding assigns codes to symbols so as to match code lengths with the probabilities of the symbols. Typically, these entropy encoders are used to compress data by replacing symbols represented by equal-length codes with symbols represented by codes where the length of each code word is proportional to the negative logarithm of the probability. Therefore, the most common symbols use the shortest codes.

2. Privacy in Arithmetic Coding

Basically, the arithmetic coding, a kind of statistical methods, operates by encoding symbols one at a time. The symbols are then encoded into variable length output codes. The length of the output code varies based on the probability or frequency of the symbol. Low probability symbols are encoded using many bits, and high probability symbols are encoded using fewer bits [3], [4]. In arithmetic coding, the codeword is a floating-point number between 0 and 1. The bigger the input size, the number of digits in the output becomes more. Like Huffman coding, the fixed model arithmetic coding is a two-pass algorithm, where the first pass computes the frequency and generates the frequency table; and the second pass does the actual compression. An input string after compression is represented by an interval of real numbers between 0 and 1. The range of the interval is initially defined by two values, high and low, which are equal to 1 and 0 respectively. The interval is successively subdivided as and when each new source symbol is encoded.

Consider the example shown in Table 1. Here, it is assumed that the source string consists of only five symbols a, b, c, d and *, where the * is referred as the termination symbol. Now, the probability of occurrence P_j for each symbol j can be calculated as

$$P_j = \frac{N_j}{\sum_{i=1}^{n} N_i}$$

Where Ni is the number of occurrence of the i^{th} symbol and n is the total no of symbols.

This leads to the overall interval range from 0 to 1, and each source symbol occupies a subinterval in that range according to its probability. For instance, the symbol b occupies the subinterval range between 0.7 and 0.9. One should note that the determination of subinterval is based on the cumulative frequency [5].

Table 1: Symbol table and their associated frequency

Symbol	*Symbol ordering*	*Probability*	*Cumulative Frequency*
a	1	0.1	0.9
b	2	0.2	0.7
c	3	0.1	0.6
d	4	0.5	0.1
*	5	0.1	0

The security in the fixed-model arithmetic coding is incorporated either by randomly ordering the symbols or by shuffling the frequencies (Bergen and Hogan 2002).

Now, encoding the given string is due to the following.

Let R(c,j), L(c,j) and H(c,j) are the range, encoding low and encoding high values of the symbol c, whose position in the string is j. Let L (#, j) and H(#, j) denotes the encoding low and high value for the character positioned at j in the given string. Then the range of a symbol c is derived as,

$$R_{(c,j)} = \begin{cases} 1 & \text{if } j = 1 \\ H_{(\#,j-1)} - L_{(\#,j-1)} & \text{otherwise} \end{cases}$$

Here, the encoding low and high values are derived as,

$$L_{(c,j)} = \begin{cases} 0 & \text{if } j = 1 \\ L_{(\#,j-1)} + R_{(\#,j-1)} * lc & \text{otherwise} \end{cases}$$

$$H_{(c,j)} = \begin{cases} 1 & \text{if } j = 1 \\ L_{(\#,j-1)} + R_{(c,j)} * hc & \text{otherwise} \end{cases}$$

Where 'lc' and 'hc' are denoted by the Low and high value of the sub-range of the symbol.

The formal description of this method is described in Algorithm 1. This can be summarized as the following pseudo sequence as in the following code. This can be implemented in any kind of programming language in the system level simulation.

Algorithm 1 (Fixed-point Arithmetic Encoding).

INPUT A string and the corresponding symbol table.
Output: Encoded value.
Begin
L = 0.0;
H = 1.0;
While (not end-of-string) do
Begin
Select next input symbol c;
R = H – L;
H = L + R * hc;
L = L + R * lc;
End
Return L;
End.

In the given example, for the string ARITHMETIC, the encoded value is 0.0757451536 which will be transmitted to the other end. The decoding is just the reverse process of encoding. The procedure for fixed-point arithmetic decoding is described in Algorithm 2.

Algorithm2 (Fixed-point Arithmetic Decoding)

Input: The encoded value and the respective symbol table.
Output: Original string.
Begin
L = 0.0;
H = 1.0;
Repeat
Value = (value – L / (H-L);
Find the symbol c such that lc ? value < hc;
Print c;
R = H – L;
H = L + R * hc;
L = L + R * lc;
Until c is the termination Symbol.
End.

Algorithm 2 terminates when the detected symbol is the termination symbol. Carrying the string length can also be used for detecting the end of string. It is to be noted that the termination symbol is not considered in the given example. The decoding process for the chosen example is illustrated in Figure 2.

Figure 2: Fixed-Point Arithmetic Decoding for the String Arithmetic

Value	Range	Low	High	Return Symbol
0.075745153600	0.1	0.0	0.1	A
0.757451536000	0.1	0.7	0.8	R
0.574515360000	0.2	0.4	0.6	I
0.872576800000	0.2	0.8	1.0	T
0.362884000000	0.1	0.3	0.4	H
0.628840000000	0.1	0.6	0.7	M
0.288400000002	0.1	0.2	0.3	E
0.884000000024	0.2	0.8	1.0	T
0.420000000120	0.2	0.4	0.6	I
0.100000000598	0.1	0.1	0.2	C

Thus, the string ARITHMETIC is decoded using this procedure.

3. Proposed Method for Security Enhancement

Arithmetic coding provides data security and compression. In fixed model arithmetic coding security is inbuilt by shuffling the symbol ordering or arbitrarily choosing the frequency of occurrence. Security is enhanced by randomly assigning probabilities to the frequency of occurrence using pseudo random numbers and finding the product between frequency of occurrence of the character and randomly selected probabilities, and by not altering the frequency of occurrence and symbol ordering [6]. The advantage is the source symbol table and the associated frequency (the model) does not specify the frequency of occurrence of each alphabet. The secrecy lies in the seed value of the random algorithm. The most widely used technique for pseudorandom number generation is an algorithm proposed by Lehmer (1951); this is known as the linear congruent method.

The inputs to the algorithm are the following;

m - the modulus m > 0
a - the multiplier 0 ? a < m
c - the increment 0 ? c < m
X0 - the starting value or seed 0 ? X0 < m

The sequence of random numbers Xn's is obtained via the

following iterative Equation.

$$Xn+1 = (a\ Xn + c) \bmod m.$$

Suppose m, a, c and X0 are integers, then this technique will produce a sequence of integers such that each integer will be within the range of 0? Xn < m. The selection values for a, c and m is critical in developing a good random number generator. Value of m near to or equal to 231 is chosen to produce a long series of distinct random numbers. To generalize the frequency of occurrence of each alphabet is calculated with the probability of occurrence of each symbol in the message and that has to be encoded. A randomly generated probability is multiplied with the probability of occurrence to generate the range. The secret key is the seed value of the random number. This is illustrated with the same example string ARITHMETIC in Figure 3.

Figure 3: Symbol table with Randomly Assigned Probabilities

Symbol	Probability P	Random Probability RP	P*RF	$P_j = \frac{P_j * RP_j}{\sum_{i=1}^{n} P_i * RP_i}$
A	.1	0.326072	0.0326072	0.265089488
C	.1	0.044391	0.0044391	0.0044391
E	.1	0.040520	0.0040520	0.032941884
H	.1	0.012341	0.0012341	0.010032966
I	.2	0.011572	0.0023144	0.018815571
M	.1	0.058361	0.0058361	0.047446231
R	.1	0.288272	0.0288272	0.234358905
T	.2	0.218472	0.0436944	0.355226038

The encoder described in Algorithm 1 can be followed for compression process. The compression process is the same as that of the arithmetic coding. The above process hides the frequency of occurrence and the calculation becomes impossible without knowing the seed value of pseudorandom number generators [8]. In the reverse process the frequency of occurrence is obtained by dividing the product of frequency and random probability by the randomly generated value to get the frequency of occurrence.

In general, the symbols are in American standard code for information interchange (ASCII) order, and the frequency values are chosen to match the occurrence of the symbol that occurs in the string that has to be compressed. The frequency of occurrence of each symbol is equal to the sum of the product of frequency of occurrence and the chosen random probability (R). Now, the probability of occurrence Pj for each symbol j can be calculated as follows by incorporating random number to the equation.

$$P_j = \frac{P_j * RP_j}{\sum_{i=1}^{n} P_i * RP_i}$$

Where Pi*RPi is the random probability of occurrence of i^{th} symbol and n is the number of symbols present in the string that has to be compressed. F is the frequency and RP is the random probability. The operations performed between frequency and random probabilities are addition, multiplication and XOR. Note that XOR suits well for encryption process. The flow diagram of enhancing security in arithmetic coding is depicted in Figure 4.

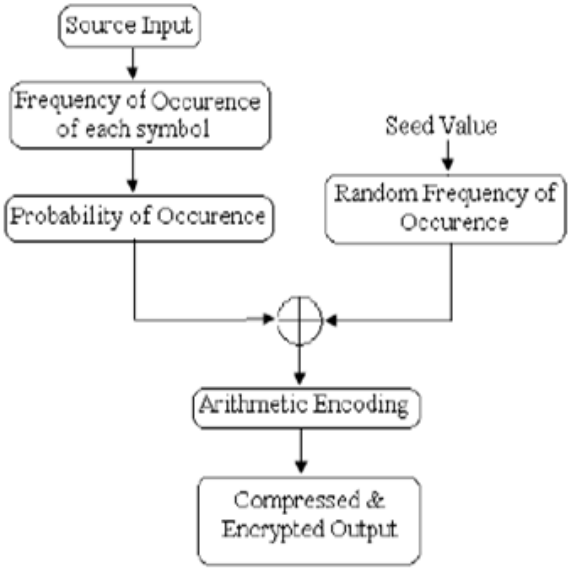

Figure 4: Flow diagram for enhancing security in Arithmetic Coding

Results and Discussion

The proposed methodology was tested using Pentium IV machine for the range of file size varies from 500 to 1500 bytes. The time taken for encryption with arithmetic encoding was compared with the time taken for simultaneous compression and encryption. Execution time taken for both compression and encryption is negligible when compared to that of encryption alone. The comparison of execution time is shown in Figure 5. The key size depends on the number of bits in the seed value of the random number. If n bits are taken as seed value then the key space is 2n. The compression ratio of arithmetic encoding is 87%. When simultaneous encryption and compression was performed there was not any significant change in compression ratio. The comparison of compression ratio is depicted in Figure 6.

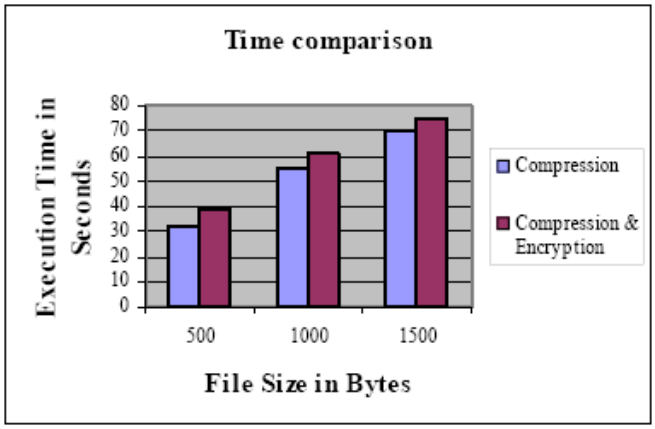

Figure 5: Time comparison between compression and simultaneous compression and encryption

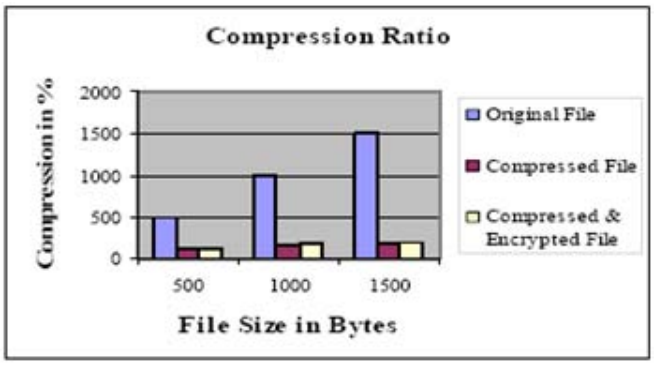

Figure 6: Compression ratios between compression and simultaneous compression and encryption

References

[1] Abramson, N. 1963. Information Theory and Coding. McGraw-Hill, New York.
[2] Apostolico, A. and Fraenkel, A. S. 1985. Robust Transmission of Unbounded Strings Using Fibonacci Representations. Tech. Rep. CS85-14, Dept. of Appl. Math., the Weizmann Institute of Science, Rehovot, Sept
[3] ARC File Archive Utility. 1986. Version 5.1. System Enhancement Associates. Wayne, N. J.
[4] Ash, R. B. 1965. Information Theory. Interscience Publishers, New York.
[5] Bentley, J. L., Sleator, D. D., Tarjan, R. E., and Wei, V. K. 1986. A Locally Adaptive Data Compression Scheme. Commun. ACM 29, 4 (Apr.), 320-330.
[6] G. Langdon and J. Rissanen, "Compression of black white images with arithmetic coding," IEEE Trans. Commun., vol. COM-29, no. 6, pp. 858–867, Jun.
[7] 1981.
[8] Brent, R., and Kung, H. T. 1978. Fast Algorithms for Manipulating Formal Power Series. J. ACM 25, 4 (Oct.), 581-595.
[9] I.H Witten, R.M Neal and J.G Cleary, Arithmetic coding for data compression, Commun. Assoc. Comput. March, 30(6) (1987) 520-540.
[10] C.B. Jones, An efficient coding system for long source sequences, IEEE Trans. Inf. Theory, vol IT -27 (1981) 280-291