AN EFFICIENT SECURE UNIVERSAL BLOCK SOURCE CODING ALGORITHM FOR INTEGERS

*Mahdi Nangir

Department of Electrical Engineering, Khaje Nasir University of Technology, Tehran, Iran

*Author for Correspondence

ABSTRACT

In this paper we present a new universal algorithm to encode the sources totally and without loss, the alphabet of output symbols is supposed the source of integer. This algorithm provides the security of system and performs the affairs of coding without using of sources statistics. Here we show that the average length of the dedicated code words is shorter than when the sequence output numbers are encoded one by one or block by block and also our presented method gives a kind of security to the system because we use some of encoding characteristic as the encryption key.

Keywords: Universal Source Coding (Data Compression), Block Source Coding, Elias Coding Schemes, Omega Coding, Secure Source Coding, Redundancy

INTRODUCTION

In recent decades, researchers of information theory, coding and cryptography fields have conducted a huge flurry of attempts toward combining algorithms to receive joint approaches. Hence, design and implementation of efficient algorithms are under main consideration when they achieve many of the following goals together: compression, error correction and security. One of the major interesting methods is devising algorithms which effectively provide both the data compression and security which has been called Joint Encryption-Source Coding (JESC) scheme.

A verity of algorithms and schemes are proposed in each categories, like the methods in (Merhav, 2006), (Kang and Liu, 2012) and (Sinaie and Vakili, 2010), using some techniques like permutation, channel coding or source coding based algorithms to achieve the compression, error correction or security together.

Elias proposed Gamma; Delta and Omega algorithms to represent positive integers which used memoryless encoding structures, all of his algorithms are universal. Each codeword in these algorithms have some attachment portions to the binary representation of integers that give capability of being uniquely decodable or instantaneous to the code. Omega encoding algorithm is the shortest one between Elias algorithms in the sense of expected codeword length over the most of probability distributions and has recursive structure which means codeword portions are obtained from each other recursively (Elias, 1975).

Foster introduces a block source coding scheme for universally encoding of positive integers (Foster et al., 2002) which encodes the source sequences block-by-block using Omega scheme (Elias, 1975). On the other hand, a recursive algorithm to encode integers with the expected code length shorter than Omega over some special probability distributions have been introduced in (Nangir et al., 2015) which uses Fibonacci representation (Apostolico and Fraenkel, 1987) in some portions of the assigned code words to the integers.

In this work, we introduce a novel algorithm on the category of JESC. The proposed method is comprised of a universal lossless block compression function and an inherent mechanism for hidden encryption of the source sequences. We aim to securely encode output sequences of a discrete information source so that the code words not only are uniquely decodable, but also are an instantaneous set of binary sequences (Cover and Thomas, 2006), we propose a new lossless block source algorithm for universal coding of positive integers which is supposed to be the source’s output alphabet without using any knowledge of the source statistics (Salomon, 2007), (Davisson, 1973). It is note to worth that the proposed scheme has a memory structure because each codeword is allocated based on all the symbols of the block. There are
some universal encoding schemes in the literature using the memory in the encoding procedure (Jalali et al., 2010), (Beirami and Fekri, 2012). Additionally, the proposed algorithm uses the count and location of small numbers of the blocks to achieve the secrecy. Algorithms encode integers one-by-one or block-by-block; we show that this scheme has shorter expected codeword length than both of these manners.

Our proposed algorithm has two important differences with works which are said in above, one of them is that our algorithm is providing secrecy by using some parameters of block coding function which is applied to output data stream of source. Another one is achieving efficient compression over all probability distributions due to have no small positive numbers in blocks, encoding 1s of blocks which have the most probability mass distribution according to the assumption is the challenging part of algorithms as we see in works like (Foster et al., 2002) and (Nangir et al., 2015). Note that these two advantages in our algorithm are intertwined with each other to give more compression and secrecy.

This paper is organized as follows. In Section 2 we present preliminaries and problem definitions. Proposed coding scheme will be seen in section 3, block source coding and construction of super code words is introduced in this section, and we bring up the main idea of block source coding in this section, key extraction and securing process is presented in this section, in Section 4 we will see its detail. In Section 5 the performance of our proposed coding scheme will be analyzed and compared with the shortest code of Elias algorithms, Omega scheme. Finally conclusions are drawn in Section 6.

Preliminaries and Problem Definition

Our general problem here is coding of integers universally, meaning that the encoder and decode don’t have any knowledge of source statistic and probability distribution on integers that are supposed source alphabet so they don’t get any usage of it, (Davisson, 1973) and (Andreassen, 2001).

It is supposed that the information source is discrete, stationary and without memory. Additionally the source probability distribution is some symbols of integers series, this is a logic and applicable in information sources because in the sources such as image the numbers of source alphabet that is pixel intensities is large, so much that with good approximation they could be illimitable and would be in one by one correspondence with integers serious.

Suppose that n is the length of each block from integers. The only assumption in universal coding algorithms has about probability distribution of symbols is the unascending feature of distribution, like the assumptions that are in (Elias, 1975)-(Knuth, 1973):

\[ P(k) \geq P(k + 1) \quad \forall k \in \mathbb{N} \]  \hspace{1cm} (1)

In which \( P(k) \) is probability distribution on integers \( \mathbb{N} = \{1, 2, 3, \ldots\} \). Additionally it is supposed that the source symbols are produced independently and identically distribution (iid). If it is supposed that the source alphabet is integer’s series is only because of simplicity, because the source output symbols alphabet is supposed illimitable and there is one by one correspondence among each discrete illimitable collection with integer’s collection.

For simplicity, the presented algorithm is applied on blocks with fixed length separately like (Foster et al., 2002). Also it is possible to analyze source output sequence to the blocks with variable length and to achieve to high compression and security advantages. In non-block structures because of the assumption of decreasing feature of probability distribution, the codeword length is decreasing function, also this feature in codeword length in block structure is true but in block structures we don’t have separate codeword, we would have a super codeword that is dedicated to blocks. The comparison of compression algorithms is performed on the basis of their complexity and average codeword length, the normalized redundancy is the illustrator of each algorithm average code length that is defined like below for p probability distribution (Han and Kobayashi, 2002):

\[ R_n(p) = \frac{EL_n(p) - H_n(p)}{H_n(p)} \]  \hspace{1cm} (2)

In which \( EL_n(p) \) and \( H_n(p) \) are showing the average length of dedicated codes to each source block and entropy that is decomposed to the blocks with length of n. Indexes shows that all calculations are
Research Article

performed on the blocks with the length of \( n \). According to first theorem of Shannon \( R_n(P) \geq 0 \) (Cover and Thomas, 2006) and it would be an efficient compression algorithm in which the normalized redundancy tends to zero. There are methods that achieve to the average length of Shannon's entropy (Jalali and Weissman, 2008a) and (Jalali and Weissman, 2008b) but have the disadvantages such as to be lossy or not to be universal. On the other hand our algorithm is lossless and universal that encodes the integers with average codeword length near entropy on the all probability distributions.

It is exploited from unary coding scheme in some part of our proposed algorithm, which is a simple universal coding algorithm, the unary code of \( m \) consists of \( m - 1 \) zero bits terminated with ‘1’ (Fiete and Seung, 2007). For example unary code of 6 is: ‘000001’.

One of the universal coding algorithms for integers that are robust against the channel errors is Fibonacci algorithm that uses Fibonacci sequence numbers to dedicate the codeword to integers. In this algorithm the integers is written like series of Fibonacci numbers that there shouldn’t be any repetitive number in series, and this demonstrates that this kind of depiction is unique and there would not be any two consecutive numbers from Fibonacci sequence in series (Apostolico and Fraenkel, 1987) and (Fraenkel and Klein, 1996). In table number 1 we see some primary numbers from Fibonacci sequence; this sequence is produced by below recursive equation:

\[
F_1 = 1, \quad F_2 = 2, \quad F_n = F_{n-1} + F_{n-2} \quad \forall n \geq 3
\]  

### Table 1: Fibonacci numbers

<table>
<thead>
<tr>
<th></th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>F4</th>
<th>F5</th>
<th>F6</th>
<th>F7</th>
<th>F8</th>
<th>F9</th>
<th>F10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>13</td>
<td>21</td>
<td>34</td>
<td>55</td>
<td>89</td>
</tr>
</tbody>
</table>

To assign a codeword to the integer \( N \), first the largest Fibonacci number that is equal to or less than \( N \) is determined, let it is \( F_n \), the \( i \)'th Fibonacci number. A “1” is placed as the \( i \)'th bit in the codeword. With subtracting \( F_i \) from \( N \) and repeating the previous steps to the number \( N - F_i \) this procedure continues. This procedure is repeated until we reach to 0. Finally, a “1” bit is added to the right side of the codeword, which means the codeword is ended. The interesting property of Fibonacci code words is containing no adjacent 1’s in the codeword except at the end of codeword (Knuth, 1973), this property is used in decoding algorithm of Fibonacci scheme. Here \( F(n) \) or \( F_n \) denotes Fibonacci codeword of the integer \( n \).

For example, the Fibonacci codeword of 2015 is

\[
F(2015) = 0101001000010011
\]

Since \( 2015 = F_2 + F_5 + F_8 + F_{13} + F_{16} \), in which \( F_i \) shows the \( i \)'th element in Fibonacci sequence.

**Proposed Coding Scheme**

In this part the presented algorithm is described in detail. Before describing about the steps of algorithm we need to talk about an idea that is relevant with dedication of super codeword to blocks.

**Decomposing Source Output to Blocks**

The source coding algorithms follow different strategies with the purpose of the data compression by average codeword length near to Shannon’s entropy, one of the most common ways of source output sequence coding is symbol by symbol or separately, that is called compression without memory. Another way and strategy is symbols coding by the structures that have memory, an example of this structure is a method in which source output sequence is decomposed to some blocks and a super codeword is dedicated for each block, the dedicated super codeword is dependent on all symbols of that block obviously. We can design a variety of algorithms with memory where their performance is better than memoryless algorithms from the average code length point of view. Moreover, it is clear that memoryless algorithms have less complexity than with memory algorithms.

It is supposed that the block sequence resulted from the analysis of source output sequence is \( B_1, B_2, B_3, \ldots \) and the super code words dedicated to them are \( SC(B_1), SC(B_2), SC(B_3), \ldots \), the working way of our algorithm
is like that after analyzing the source output sequence to blocks by using of some parameters and characteristics of each block, the super code word relevant to that block would be achieved, we will describe the continuation of the process in detail.

**Location and Count of Small Numbers**

Now we describe the process of code dedication in detail. Suppose that information source in every period of time $T$ produces an integer and $X_1, X_2, ..., X_n$ is the source illimitable output sequence that is $X_i \in \mathbb{N}$. This sequence is decomposed to the blocks with fixed length like below:

$$B_1 = (X_1, X_2, ..., X_n), B_2 = (X_{n+1}, X_{n+2}, ..., X_{2n}), ...$$

Our algorithm dedicates the super code words to blocks in a multistep process.

The first step: in this step the encoder saves the unary code of block symbols in a table vertically, for example if the current block is $(2,5,1,3,1)$, the unary code words that are written vertically would be like Table 2.

**Table 2: Unary codes of current block (look-up table)**

<table>
<thead>
<tr>
<th>2</th>
<th>5</th>
<th>4</th>
<th>1</th>
<th>3</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

It is obvious that the $k$’th raw of this table depicts the numbers of $k$ in each block, so the sum of first row of this table is equal with the number of 1’s in each block that according to assumption is the most probable symbol, and should has the shortest codeword length, the important point that we will use it.

Suppose that $Y$ is the sum of 1sin first raw of table that is also representing the number of 1 of the block that we are going to code it, the number of all possible cases for the placing of these 1s in a block with length $n$ is equal with $\binom{n}{Y}$. In the next step, we code the integer’s algorithm of the block with a special code. In total in this step every universal algorithm for symbol by symbol coding of the integers is applicable.

**Table 3: Code words of some integers based on second step**

<table>
<thead>
<tr>
<th>Integer number</th>
<th>The proposed code word</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>0011001</td>
</tr>
<tr>
<td>23</td>
<td>10110111</td>
</tr>
<tr>
<td>53</td>
<td>0001110101</td>
</tr>
<tr>
<td>78</td>
<td>10011001110</td>
</tr>
<tr>
<td>1000</td>
<td>100011111101000</td>
</tr>
<tr>
<td>2012</td>
<td>0100111111011100</td>
</tr>
</tbody>
</table>

The second step: in this step an efficient recursive algorithm is used that it has three sub-steps, this algorithm is applied for the integer numbers $k > 1$ in current block:

1. Write the standard binary representation of the positive integer number and remove its most significant bit, e.g., we write 0001 instead of 17.

2. Count the number of bits that is obtained in the first step, for an integer $k$ we has $\lceil \log_2 k \rceil$ bits exactly.

3. Attach the Fibonacci code of $\lceil \log_2 k \rceil$ to the left side of the binary string we obtained in the first step.

**Example 1:** The integer number 17 has 4 bits in the first step and hence the Fibonacci codeword for the integer 4 is $F(4) = 1011$. We attach 1011 to the left side of 0001 and finally obtain the codeword...
In Table 3, we provide the code words of some positive integers based on the proposed scheme.

In this algorithm there is no dedicated codeword to the number 1, on the other hand the codeword of number 1 would be void, it should be noticed that according to the structure of block coding that we described before, this lack of dedication of codeword of the number 1 doesn’t cause ant problem at all because we want to encode the \( k \geq 2 \) integers in every block.

Now we continue the description of the algorithm. In every block there is a series of 1s and \( k \geq 2 \) integers. The way of creating super code words is like that we disregard (delete) the 1s in the blocks and get the new blocks \( B' \) and apply the numbers of the new blocks of the mentioned symbol by symbol coding algorithm. Look at this example:

**Example 2:** Let assume \( B = (2, 5, 4, 1, 3, 1) \), then we have \( B' = (2, 5, 4, 3) \). To assign a super codeword to the block \( B \), the codes obtained from the stage 2 of the proposed algorithm are serially concatenated. Thus, the corresponding super codeword is \( SC(B) = (c(2), c(5), c(4), c(3)) \). Therefore, the associated super codeword of the encoder is \( SC(B) = (011, 00011, 1011, 0011) \).

In figure 1, the construction of super codeword from source output sequence is shown in detail, notice that two keys of 1 and 2 are used to dedicate the super code words to blocks. The process of gaining the key from source sequence is described in next sub-section.

**Figure 1: Construction of super code words**

**Key Extraction for Secure Coding Procedure**

As it was mentioned \( Y \) and \( \binom{n}{Y} \) are two important parameters in our coding algorithm. The binary depiction of these numbers is used as encryption key in presented algorithm. Due to the key exchange our encryption algorithm belongs to the symmetric-key encryption category (Denning, 1982), (Ahmad et al., 2015). As we see in figure 2, two keys that are binary depiction of \( Y \) and \( \binom{n}{Y} \) which are used for joint decoding and deciphering in presented algorithm simultaneously.

key 1=binary presentation of \( Y \)

key 2=binary presentation of \( \binom{n}{Y} \) \( \tag{4} \)

In figure 2, we show the block diagram of the proposed algorithm. Two keys which are binary presentations of \( Y \) and \( \binom{n}{Y} \), respectively, are employed for joint decoding and deciphering.
As we see in figure 2 the discrete sequence \(X_1, X_2, X_3,\ldots\) is the source output that is entered to block encoder and the super code words are produced like it was told. The encryption keys are extracted from \(X_1, X_2, X_3,\ldots\) sequence and are sent to the receiver by a secure channel till could be used in simultaneous process of decoding and deciphering.

**Decoding and Deciphering Procedure**

The decoding scheme of a universal coding scheme should be universal too because the decoder doesn’t have any information about source symbol statistics and probability distribution (Ordentlich et al., 2008). Here the number and the place of 1s are also unknown for the decoder at first and are received as key which is used not only decoding but also in deciphering for the super code words.

In this section the joint decoding and deciphering are described. As you see in the figure the receiver block has three entrance terminals that are from up to down like this:

1- \(SC(B_1), SC(B_2), SC(B_3),\ldots\)
2- key 1
3- key 2

Because these three entrances enter to the receiver block from different terminals, this block would differentiate between them naturally, all these entrances are binary sequences. From these three entrances we want to discover the \(X_1, X_2, X_3,\ldots\) sequence; we explain the algorithm in detail below.

First of all the terminal 2 that the \(Y\) binary depiction is available in it, get the amount of this parameter. After that the amount of \(n\) parameter is calculable from terminal 3 because if we suppose the binary sequence equivalent in terminal 3 as \(m\) we would have \(m \leq \binom{n}{Y}\), in fact \(n\) is the smallest number that could get this inequality. The simplest way to get \(n\) is the trial and error method.

After these steps we are prepared till to decode the super words sequence. Since \(n\) and \(\binom{n}{Y}\) parameters are available, the number and place of 1s are known in sequence for the decoder. At first the 1s are put in relevant place in each block and then the super code words in terminal 1 of \(k \geq 2\) integers are decoded and will place in empty places (the places in which there is no number 1) of each block till the all sent sequence are decoded. Notice that in decoding of super code words in terminal 1 of the receiver we should use the decoding of the coding scheme that was explained in step 2, that we will explain it in detail in continue, this decoding algorithm is very similar to that decoding algorithm which is used in (Nangir et al., 2015).

Imagine the binary sequence of received super code words in terminal 1, the bit sequence are read from first till we arrive to the first sub-stream “11” (two consecutive bit 1), then by using of decoding algorithm of Fibonacci method (Apostolico and Fraenkel, 1987) we decode till to decode that part of sequence and get the \(K\) integer (the resulted number from Fibonacci decoding), then we read \(K\) bitsafter...
“11” sub-stream attaching it a 1 bit as most significant bit (MSB) and introduce it as the sent integer. In continue we will depict an example to make clearer the process of joint decoding and deciphering procedure.

Example 3: suppose that below binary streams were received in three terminals of receiver block:
- Terminal 1: 0110001110110011
- Terminal 2: 10
- Terminal 3: 1110

As it was explained we have $Y=2$ from terminal 2, after getting $Y$ we can calculate $n$ like this that $n$ is the smallest integer that would true in $\binom{n}{Y} = \binom{n}{2} \geq 14 = m$ inequality, the $m=14$ is got from terminal 3. From this inequality $n=6$ is obtained. Now from 1110 stream in terminal 3 decoder find out that this is correspond this position of “1”s in six-tuple: (−, −, −, 1, −, 1).

Finally from the binary sequence in terminal 1 and by using the second step decoding process we can achieve to the sextet blanks. During this process the 0110001110110011 is decoded to (2, 5, 4, 3), by combining these numbers with data about number and the place of numbers 1 that now are available we depict the sent sequence like this $(X_1, X_2, X_3, X_4, X_5, X_6) = (2, 5, 4, 1, 3, 1)$ and the process of joint decoding and deciphering is completed.

The arrangement and the way of imputing the binary streams of terminal 3 to the place of 1s in blocks of n integer give more freedom and visibility to us in designing powerful keys for system. This arrangement and imputation is obviously hidden for the impermissible and unknown user and couldn’t find the sent sequence by receiving the super code words sequence. In last example the binary filed imputation of terminal 3 to the place of 1s is like this strategy:

Imagine two binary stream $b_1$ and $b_2$ from terminal 3, if the corresponding n-tuple for placing 1s are $B_1$ and $B_2$, we suppose them the place of 1s in n-tuple. Our imputation strategy is like below:

$$b_1 > b_2 \quad (5).$$

If and only if

$$\sum \text{index of "1" in } B_1 > \sum \text{index of "1" in } B_2 \quad (6).$$

It is obvious that we could have different strategies and formulas that their results would be variable keys and this cause that the breaking the cipher of this system wouldn’t be easy. In next section we will discuss the algorithm performance in the sense of the average codeword length.

**Performance Analysis of Proposed Algorithm**

Our presented algorithm gets a smaller average codeword length than Omega Elias scheme over all probability distributions that between three Elias’s algorithms has the smallest average codeword length, we will affirm it. Moreover the security operation of presented algorithm is powerful, to break the code of this algorithm when the attacker is aware of the n, block length, for number and placing of the 1s of the blocks there are $2^n$ ways, so the code breaking of this algorithm contains obvious complexes that are more than polynomial complexes in the case of calculation, which is in NP complex categories.

Some code words of small integers are presented in table 4, in the sense of compression the average codeword length is the standard comparison between the codes.

**Lemma 1:** For $n \geq 4$ the Fibonacci coding scheme has shorter (it means $\leq$) codeword length than Omega Elias coding scheme which have shortest expected codeword length between three algorithms of Elias (Sayood, 2003).

**Lemma 2:** For $n \geq 2$ the Second Stage coding scheme has shorter (it means $\leq$) codeword length than Omega Elias coding scheme.

**Proof:** Because we use Fibonacci code words in the left portion of codeword in Second stage scheme and recursive structure of Omega Elias scheme, for $n \geq 2$ this lemma is strongly true using lemma 1 (For
length of Fibonacci code words no closed form expression is presented up to now, due to this we note that simulation results confirm lemma).

**Theorem 1:** Our proposed algorithm achieves shorter expected codeword length than Elias Omega scheme over all probability distributions.

**Proof:** suppose that the length of Omega coding scheme for positive integer \( n \) is \( \omega(n) \). Then the expected codeword length of Omega coding is as follows

\[
E[\omega] = \sum_{i=1}^{\infty} p_i \omega(i) \tag{7}
\]

Also suppose that the length of our coding scheme for positive integer \( n \) is \( c(n) \). Then the expected codeword length of our coding is as follows (notice that \( c(n) \) is codeword length which obtained from stage 2, so \( n \geq 2 \) )

\[
E[c] = \sum_{i=2}^{\infty} p_i c(i) \tag{8}
\]

Omega scheme is a recursive algorithm, it means that if we eliminate the last portion of code words in Omega codeword without the last “0” bit of it, we achieve to a sequence of bits that is an Omega codeword for another positive integer too.

From lemma 2 we conclude that for \( n \geq 2 \) the codeword length which obtained from second stage has shorter length than Omega coding scheme length, because the left side portion of codeword arises from Fibonacci codeword i.e.,

\[
\forall n \geq 2: \quad c(n) \leq \omega(n) \tag{9}
\]

An expectation over probability distribution \( \{ p_i \} \) which is unknown to us results,

\[
E[c] = \sum_{i=2}^{\infty} p_i c(i) \leq \sum_{i=2}^{\infty} p_i \omega(i) < E[\omega] = \sum_{i=1}^{\infty} p_i \omega(i) \tag{10}
\]

Note that in our algorithm no codeword is assigned to the 1s in blocks. This inequality completes the proof.

**Table 4: Code words of 1 to 14 based on Fibonacci, second stage and Omega Elias schemes**

<table>
<thead>
<tr>
<th>Integer Number</th>
<th>Fibonacci Scheme</th>
<th>Second Stage Scheme</th>
<th>Omega Elias Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>011</td>
<td>110</td>
<td>10 0</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
<td>111</td>
<td>11 0</td>
</tr>
<tr>
<td>4</td>
<td>1011</td>
<td>01100</td>
<td>10 100 0</td>
</tr>
<tr>
<td>5</td>
<td>00011</td>
<td>01101</td>
<td>10 101 0</td>
</tr>
<tr>
<td>6</td>
<td>10011</td>
<td>01110</td>
<td>10 110 0</td>
</tr>
<tr>
<td>7</td>
<td>01011</td>
<td>01111</td>
<td>10 111 0</td>
</tr>
<tr>
<td>8</td>
<td>000011</td>
<td>0011000</td>
<td>11 1000 0</td>
</tr>
<tr>
<td>9</td>
<td>100011</td>
<td>0011001</td>
<td>11 1001 0</td>
</tr>
<tr>
<td>10</td>
<td>010011</td>
<td>0011010</td>
<td>11 1010 0</td>
</tr>
<tr>
<td>11</td>
<td>001011</td>
<td>0011011</td>
<td>11 1011 0</td>
</tr>
<tr>
<td>12</td>
<td>101011</td>
<td>0011100</td>
<td>11 1100 0</td>
</tr>
<tr>
<td>13</td>
<td>00000111</td>
<td>0011101</td>
<td>11 1101 0</td>
</tr>
<tr>
<td>14</td>
<td>10000111</td>
<td>00111110</td>
<td>11 1110 0</td>
</tr>
</tbody>
</table>
Conclusion

A new JESC algorithm was presented that encodes the integers securely and universally. In the algorithm process the analysis idea of source output sequence to blocks is used which is supposed as a sequence of integers. Instead of dedicating codeword to each integer we dedicate a super codeword to each block. It was showed that by decomposing the source output sequence to the series of blocks we can use the produced parameters, we used the parameters in extracting encryption keys and securing the coding algorithm. Finally the process of presented algorithm is analyzed and compared about its efficiency in universal compression. The amount of security of the presented algorithm is also analyzed in the sense of the calculation the complexity of code breaking by the enemy.

REFERENCES


