International Journal of Advanced Studies in Engineering and Scientific Inventions | IJASESI
Hard Print: 2354-4171 | Online: 2354-418X | Vol. 4, No. 1 May, 2017

New Lempel-Ziv-Welch Fault Tolerant Data Compression and Encryption Scheme

[1]Abdul-Barik Alhassan, [2]Kazeem Alagbe Gbolagade, & [3]Edem Kwedzo Bankas
[1&3]Department of Computer Science
University for Development Studies, Navrongo, Ghana
[2]Kwara State University, Malete, Ilorin, Kwara State, Nigeria

Corresponding Author: Abdul-Barik Alhassan

Abstract

Data compression algorithms are designed to maximize storage space and bandwidth in transmission via data communication channels of limited bandwidth. Currently, most of our communications or business transactions are now done using some form of network or the internet, it is therefore essentially relevant to ensure data security and integrity across noisy communication channels or less reliable storage devices. This can be achieved by an efficient data compression, encryption, and error detection and correction scheme. Error detection involves the sending of additional data or information to detect and reject incorrect data whiles error correction involves the addition of data or information to allow for correction and acceptance of data. Several error detection and correction schemes exist. In Residue Number System (RNS), most error detection and correction schemes involves the use of additional moduli termed redundant moduli, for error detection and correction. In this paper, RNS has been applied to the Lempel-Ziv-Welch (LZW) data compression algorithm using the moduli set $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$, resulting in a new LZW-RNS compression and encryption scheme. Two redundant moduli, $\{2^{2n} - 3\}$ and $\{2^{2n} + 1\}$are added to detect and correct errors in encrypted and compressed data as applied in Redundant Residue Number System (RRNS). The proposed scheme also allows for four bit stream residual archiving or transmission of data. The encoder and decoder pairs have also been constrained to work for only even *n* numbers, as an additional security measure. Implementation in MatLab reveals that the secured fault-tolerant scheme performs better in detecting and correcting errors than the LZW algorithm and other known similar state of the art schemes.

Keywords: *Compression, Correction, Detection, Encryption, Error.*

http://internationalpolicybrief.org/journals/international-scientific-research-consortium-journals/intl-journal-of-adv-studies-in-engr-and-scientific-inventions-vol4-no1-may-2017

Background to the Study

Residue Number System (RNS) usage continues to gain popularity as a result of the fact that a great deal of computing now takes place in embedded processors. Computer chips are also now dense that full testing is cumbersome and laborious or impossible, hence the need for fault tolerance and enhancement in computational integrity of digital devices (Pahami, 2000; Omondi and Pumkumar, 2007 and Lu, 2004).

Data compression is the process of converting an input data stream into another data stream that has a smaller size (Mahoney, 2012). Various compression algorithms exist for compressing different types of file formats including text, image, sound, video or a combination of these. These algorithms are either classified as lossy or lossless, and dictionary or non-dictionary based depending on the nature of the output for a specific input (Jane and Trivedi, 2014 and Shammugasundaram and Lourdusamy, 2012).In improving the security, transmission, and storage efficiencies of various algorithms, RNS has been efficiently applied (Alhassan, Gbolagade and Bankas, 2015).

Alhassan, Saeed, and Agbedemnab (2015) proposed a scheme for improving the efficiency of the Huffman's method of data encoding where the frequency of occurrences of each character are used to generate binary codes to reduce data size and enhance security. There exist a lot of literature on improving the Lempel-Ziv-Welch (LZW) algorithm. Alhassan, Gbolagade, and Bankas (2015) also applied RNS using the traditional moduli set $\{2^n - 1, 2^n, 2^n + 1\}$ to propose a novel LZW-RNS compression scheme which showed better performance than the traditional lossless dictionary based LZW algorithm in terms of improved compression efficiency, security, and speed of execution. Kaur and Verma(2012), modified the LZW data compression algorithm as Content-based Addressable Memory (CAM) array which utilizes less bits than its ASCII code.Welch (1984) and Rodeh, Pratt, and Even (1981) summarily stated that data stored on disk or tapes or transferred over communication media in commercial computer systems generally contains significant redundancy. The research also proposed a new scheme whose principle is not found in general commercial compression methods.

Shammugasundaram and Lourdusamy, (2012) presented a comparative study of text compression algorithms where the LZW is found to be the least performing in terms of bits per compression (BPC). Mahyar (2012) noticed that building 3G using KASUMI block cipher to perform better than 2G and 2.5G networks is error prone which was dealt with by using Redundant Residue Number System (RRNS) that has inherent error correction capabilities. A method premised on modulus projection approach where the algorithm reduces considerably the computation overhead for RRNS codes decoding was proposed (Amusa and Nwoye, 2012).Data compression is very essential, and defined as a method or encoding technique which reduces data size substantially based on an existing law, of which important applications in storage systems and communication networks should be highly secured (Mahoney, 2012; Welch, 1984, and Rodeh, Pratt, and Even, 1981).

Barati, Dehgan, and Movaghar (2008), applied RRNS to data distribution for mobile systems and wireless networks based on peer-to-peer protocol. It error detection and correction capability was used together with multi-level RNS to propose a new scheme which showed better performance.

In this paper, Residue Number System (RNS) is utilized with the moduli set $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$ to the Lempel-Ziv-Welch (LZW) algorithm resulting in new efficient LZW-RNS scheme. Two redundant moduli, $\{2^{2n} - 3\}$ and $\{2^{2n} + 1\}$ are employed in order to detect and correct errors in the compressed and encrypted data.

The rest of the paper is organized into four sections. Section II involves the presentation of materials and methods such as the LZ family of algorithms, proposed algorithm, RNS and RNS conversion processes. Presentation of results and findings including the proposed encoder and decoder, performance analysis and implementation is in Section III. In Section IV, relevant conclusions are drawn from the findings whiles Section V entails the extension for future research.

Materials and Methods

Generally, compression algorithms are categorized into non-dictionary or dictionary-based, and lossy or lossless. Lossless compression algorithms allow for decoding back the original data whiles loss compression algorithms allows for an approximation of the original data (Mahoney, 2012, and Lempel and Ziv;1977, 1987). The Lempel-Ziv compression algorithm is an adaptive coding scheme where the words of the source alphabet are dynamically defined as the encoding is performed, and it is the basis for the UNIX utility compress. The LZ compression algorithm is a family of compression algorithms that was presented by Abraham Lempel and Jacob Ziv in their landmark papers in 1977 and 1978 respectively (Shammugasundaram and Lourdusamy, 2012; Ziv and Lempel, 1977, 1978).

LZW Compression Algorithm

The LZW algorithm is an improved lossless dictionary-based compression algorithm that was published by Welch in 1984. The research improved on the existed algorithm that was published by Ziv and Lempel in 1978. It is simple to implement, and has the potential for very high throughput in hardware implementations (Shammugasundaram and Lourdusamy, 2011, and Lempel and Ziv, 1977, 1978).

Residue Number System (RNS)

The representation of numbers with remainders otherwise known as RNS is defined by a basis consisting of a set of relatively prime numbers, called moduli $\{m_1, m_2, m_3 \ldots m_n\}$, where the greatest common divisor (gcd) between any two moduli is one, that is gcd $(m_i, m_j) = 1, \forall\, i \neq j$. An integer X is represented by n-tuple $(r_1, r_2, \ldots, r_n)$ in RNS where the residue $r_i = |X|_{mi}$ for $i = 1,2, \ldots, k$ and $|X|_{mi}$ is defined as $X \bmod m_i$. The Dynamic range of the RNS is given by $M = \prod_{i=1}^{n} m_i$. Applying the Chinese Remainder Theorem (CRT), an integer X be calculated from its residue digits $(r_1, r_2, \ldots, r_k)$ as follows;

$$X = \left| \left| \sum_{i=1}^{k} M_i^{-1} r_i \right|_{m_i} M_i \right|_M \qquad (1)$$

Where; $M = \prod_{i=1}^{k} mi, Mi = \frac{M}{m_i}$, and M_i^{-1} is the multiplicative inverse of M_i with respect to m_i (Mohan and Preekumar, 2007; Pahami, 2000, and Lu, 2004)

Redundant Residue Number System (RRNS)

RNS is primarily used for high-speed digital signal processing due to the modular carry free arithmetic operations. It is defined by a set of relatively prime integers $(m_1, m_2, \ldots, m_n)$ called non-redundant moduli. Error detection and correction properties are introduced by addition of redundant moduli. Thus RNS is defined by the moduli set $(m_1, m_2, \ldots, m_n, m_{n+1}, m_{n+2}, \ldots, m_k)$. Both the non-redundant and redundant moduli should be relatively prime and should satisfy the condition $(m_1, m_2, \ldots, m_n) > (m_{n+1}, m_{n+2}, \ldots, m_k)$. The total dynamic range of RRNS is given by $M_T = \prod_{i=1}^{n} m_i$. This total range [0, M_T] is divided into two adjacent intervals in the ranges defined by the non-redundant and redundant moduli. The interval [0, M) is termed the legitimate range where M= $\prod_{i=1}^{n} m_i$ and the interval [M, M_T) is termed the illegitimate range. The error detection and correction demands that we constrain the information within the legitimate range(Lu, 2004; Pahami, 2000, and Yang and Hanzo, 2001).

The Conversion Process

Convertors are used to change from one number representation to the other. A forward convertor converts from Decimal to RNS (D/R), and the reverse convertor converts from RNS to Decimal (R/D) representation respectively (Omondi and Preemkumar, 2007; Pahami, 2000, and Mohan and Preemkumar, 2007).

Forward Conversion Process for the Moduli Set $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$

The encoder pair is designed using the LZW-RNS scheme and the forward convertor to convert from binary/decimal to residue as follows (Bankas and Gbolagade, 2012; Gbolagade, Voicu, and Cotofana, 2010, and Mohan and Preemkumar, 2007);

Given the Moduli set $\{2^n, 2^n - 1, 2^n + 1, 2^{n+1} - 1\}$where $m_1 = 2^n, m_2 = 2^n - 1, m_3 = 2^n + 1$, and $m_4 = 2^{n+1} - 1$. Let $m_5 = 2^{2n} - 3$, and $m_6 = 2^{2n} + 1$, be redundant moduli for the purpose of detecting and correcting errors in the decoding process as well as securing data. The moduli set therefore is a combination of both the redundant and non-redundant $\{2^n, 2^n - 1, 2^n + 1, 2^{n+1} - 1, 2^{2n} - 3, 2^{2n} + 1\}$ with a corresponding residue set $\{r_1, r_2, r_3, r_4, r_5, r_6\}, r_i, i = \{1, 2, 3, 4, 5, 6\}$.

Forward Conversion Process

For the given moduli set, any binary number X is represented as a $4n - bit$ number as;

$X = x_{4n-1}x_{4n-2} \ldots x_1x_0$ (3)

Since $r_i = |X|_{m_i}, i = 1, 2, 3, 4, 5, 6$, the $r_i's$ are computed as follows:

$r_1 = x_{n-1}x_{n-2} \ldots x_1x_0$ (4)

Which is the $n -$ least significant bit of any number X.

In order to compute the remaining r_is, the number X is partitioned into $4n$-bit blocks as follows:

$$B_1 = \sum_{j=3n}^{4n-1} x_j 2^{j-3n}, B_2 = \sum_{j=2n}^{3n-1} x_j 2^{j-2n}, B_3 = \sum_{j=n}^{2n-1} x_j 2^{j-n} \text{ and } B_4 = \sum_{j=0}^{n-1} x_j 2^j \quad (5)$$

which implies;

$X = 2^{3n}B_1 + 2^{2n}B_2 + 2^nB_3 + B_4$ (6)

Therefore,

$r_2 = |2^{3n}B_1 + 2^{2n}B_2 + 2^nB_3 + B_4|_{2^n-1}$

$= |B_1 + B_2 + B_3 + B_4|_{2^n-1}$ (7)

$r_3 = |2^{3n}B_1 + 2^{2n}B_2 + 2^nB_3 + B_4|_{2^n+1}$

$= |-B_1 + B_2 - B_3 + B_4|_{2^n+1}$ (8)

and

$$r_4 = |2^{3n}B_1 + 2^{2n}B_2 + 2^nB_3 + B_4|_{2^{n+1}-1}$$
$$= |2^{n-2}B_1 + 2^{n-1}B_2 + 2^nB_3 + B_4|_{2^{n+1}-1} \quad (9)$$

Similarly, the redundant residues r_5 and r_6 are computed as follows:

$$r_5 = |2^{3n}B_1 + 2^{2n}B_2 + 2^nB_3 + B_4|_{2^{2n}-3}$$
$$= |2^{n+1}B_1 + 2^nB_1 + 2B_2 + B_2 + 2^nB_3 + B_4|_{2^{2n}-3} \quad (10)$$

and

$$r_6 = |2^{3n}B_1 + 2^{2n}B_2 + 2^nB_3 + B_4|_{2^{2n}+1}$$

$$= |-2^nB_1 - B_2 + 2^nB_3 + B_4|_{2^{2n}+1} \quad (11)$$

Implementation

Equations (9) – (11) can further be simplified as;

$$r_4 = |2^{n-2}B_1 + 2^{n-1}B_2 + C|_{2^{n+1}-1} \quad (12)$$
$$r_5 = |2^{n+1}B_1 + 2B_2 + C + D|_{2^{2n}-3} \quad (13)$$

and,

$$r_6 = |-2^nB_1 - B_2 + C|_{2^{2n}+1} \quad (14)$$

Where;

$$C = 2^nB_3 + B_4 = \underbrace{B_{3,n-1}B_{3,n-2}\dots B_{3,1}B_{3,0}}_{n-bit}\overbrace{00\dots0}^{n-bit} + \overbrace{00\dots0}^{n-bit}\underbrace{B_{4,n-1}B_{4,n-2}\dots B_{4,0}}_{n-bit}$$
$$= \underbrace{B_{3,n-1}B_{3,n-2}\dots B_{3,1}B_{3,0}}_{n-bit} \bowtie \underbrace{B_{4,n-1}B_{4,n-2}\dots B_{4,1}B_{4,0}}_{n-bit}$$
$$= \underbrace{B_{3,n-1}B_{3,n-2}\dots B_{3,1}B_{3,0}B_{4,n-1}B_{4,n-2}\dots B_{4,1}B_{4,0}}_{2n-bit} \quad (15)$$

and,

$$D = 2^nB_1 + B_2 = \underbrace{B_{1,n-1}B_{1,n-2}B_{1,1}B_{1,0}}_{n-bit}\overbrace{00\dots0} + \overbrace{00\dots0}\underbrace{B_{2,n-1}B_{2,n-2}B_{2,1}B_{2,0}}_{n-bit}$$
$$= \underbrace{B_{1,n-1}B_{1,n-2}B_{1,1}B_{1,0}}_{n-bit} \bowtie \underbrace{B_{2,n-1}B_{2,n-2}B_{2,1}B_{2,0}}_{n-bit}$$
$$= \underbrace{B_{1,n-1}B_{1,n-2}B_{1,1}B_{1,0}B_{2,n-1}B_{2,n-2}B_{2,1}B_{2,0}}_{2n-bit} \quad (16)$$

Architecture

The architecture for computation of the residues is realized through (6) – (8) and (12) – (16). It begins with an Operands Preparation Unit (OPPU) which prepares the operands by simply manipulating the routing of the bits of the number X into the four blocks. Equations (15) and (16) involve the joining of bits which do not require any hardware. The computation of r_1 does not also require any hardware since it is the $(n-1)$-bit least significant bit of the number X. The rest of the process requires regular Carry Save Adders (CSAs) and regular Carry Propagate Adders (CPAs) which takes inputs from the CSAs. CSAs take in three inputs whilst the CPAs take in two inputs. The schematic diagrams for the proposed forward converter are as follows:

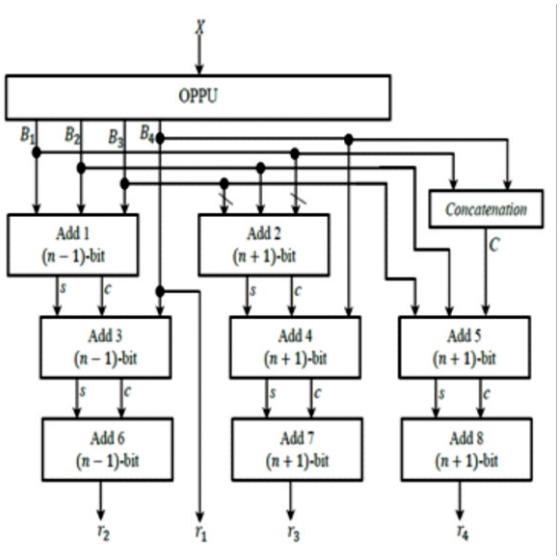

Figure 1.2: Forward Converter (RNS)

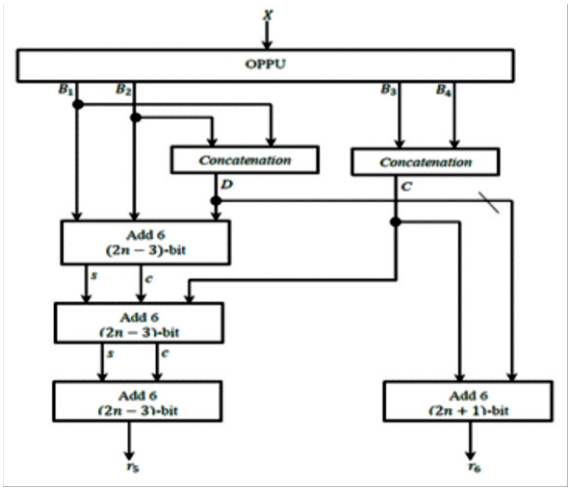

Figure 1.3: Forward Convertor (RRNS)

Numerical Illustrations

Example 1.1: Given the moduli set $\{2^n, 2^n - 1, 2^n + 1, 2^{n+1} - 1\}$, and $\{2^{2n} - 1,\ 2^{2n} + 1\}$ as redundant moduli; take $n = 2$ and consider X representing a character of a message to be encoded with ASCII character representation $X = 25$. Then, the conversion or residue set $\{r_i\}_{i=1,2,\ldots,6}$ is obtained as follows;

$25_{decimal} = 00011001_{binary}$; giving blocks $B_1 = 00$, $B_2 = 01$, $B_3 = 10$, and $B_4 = 01$.

$r_1 = n - LSB(00011001) = 01$

$r_2 = |B_1 + B_2 + B_3 + B_4|_{2^n - 1} = |00 + 01 + 10 + 01|_{11} = 01$

$r_3 = |-B_1 + B_2 - B_3 + B_4|_{2^n + 1} = |-00 + 01 - 10 + 01|_{101} = 000$

$r_4 = |2^{n-2}B_1 + 2^{n-1}B_2 + 2^n B_3 + B_4|_{2^{n+1}-1} = |2^0 00 + 2^1 01 + 2^2 10 + 01|_{111}$

$= |00 + 010 + |1000|_{111} + 01|_{111} = |00 + 010 + 001 + 01|_{111} = 100$

$r_5 = |2^{n+1}B_1 + 2^n B_1 + 2B_2 + B_2 + 2^n B_3 + B_4|_{2^{2n}-3} =$

$= |2^3 00 + 2^2 00 + 2^1 01 + 01 + 2^2 10 + 01|_{1101}$
$= |0000 + 0000 + 010 + 01 + 1000 + 01|_{1101} = 1100$
$r_6 = |-2^n B_1 - B_2 + 2^n B_3 + B_4|_{2^{2n}+1} = |2^2 00 - 01 + 2^2 10 + 01|_{10001}$
$= |0000 - 010 + 1000 + 01|_{10001} = 01000$
Thus, the residue set $\{r_i\}_{i=1,2,\ldots,6}$ is $\{01, 01, 000, 100, 1100, 01000\}$ which is equivalent to $\{1, 1, 0, 4, 12, 8\}$ in decimal.
That is, $|25|_{\{2,3,5,7,13,17\}} = \{1, 1, 0, 4, 12, 8\}$.

Reverse Conversion Process for the Moduli Set $\{2^n - 1,\ 2^n,\ 2^n + 1,\ 2^{n+1} - 1\}$
The residue to binary conversion process is done using the Chinese Remainder Theorem (CRT) according to equation (1) as follows;

Given any RNS number X, with a four residue combinations r_{ijkl}, corresponding to any moduli m_{ijkl}, then the M_{ijkl} and the M_{ijkl}^{-1} can be computed accordingly and then substituted into the CRT in equation (1) in order to get X in binary.

Example 1.2: Given any RNS number $X = \{r_1, r_2, r_3, r_4\}$ with respect to the moduli set $\{2^n, 2^n - 1, 2^n + 1, 2^{n+1} - 1\}$ in the order $m_i, i = 1, 2, 3, 4$ respectively, we have;
$M_1 = (2^{n+1} - 1)(2^{2n} - 1)$, $M_2 = 2^n(2^n + 1)(2^{n+1} - 1), M_3 = 2^n(2^n - 1)(2^{n+1} - 1)$, and$M_4 = 2^n(2^{2n} - 1)$ (17)

Their multiplicative inverses are as follows:
$\left|M_1^{-1}\right|_{m_1} = 1, \left|M_2^{-1}\right|_{m_2} = |-1|_{m_2}$, $\left|M_3^{-1}\right|_{m_3} = |-1|_{m_3}$,and$|M_4^{-1}|_{m_4} = |2^{n-1}|_{m_4}$ (18)

Thus, the RNS number can be converted back to binary as follows:

$$X = \left| \begin{matrix} (2^{n+1} - 1)(2^{2n} - 1)r_1 - 2^n(2^n + 1)(2^{n+1} - 1)r_2 \\ -2^n(2^n - 1)(2^{n+1} - 1)|(-2^{n-1})r_3|_{2^n+1} \\ +2^n(2^{2n} - 1)|(2^{n-1})r_4|_{2^{n+1}-1} \end{matrix} \right|_{2^{n-1}(2^{2n}-1)(2^{n+1}-1)} \quad (19)$$

Error Handling Mechanism
By employing the two redundant moduli $\{2^s - 3\}$and $\{2^s + 1\}$. In RRNS, a number represented with original moduli set (with number of moduli) can still be represented with the original chosen moduli and redundant moduli ((−)number of redundant moduli). The redundancy in the system allows for the reconstruction of that number by using any combinations of the moduli at the receiver and such RRNS (,)code has capability of simultaneously detecting s residue digit errors and correcting t random residue digit errors, if and only if + ≤ (−) (Alhassan, Gbolagade, and Bankas, 2015; Omondi and Preemkumar, 2007, and Yang and Hanzo, 2001).

Example 1.3: To show the use of the CRT to recover a number from its residue digits. The number 25 can be represented using the moduli set (4, 3, 5, and 7) as (1, 1, 0, 4). To convert (1, 1, 0, 4) back to a decimal representation, the CRT is applied as follows: M = 4 x 3 x 5 x 7 = 420
= $\frac{420}{}$= (105, 140, 84, 60)

The multiplicative inverses are obtained as follows:

$|105|_4 = 1$, $|1|_4 = 1$, therefore $_1^{-1} = 1$
$|140|_3 = 2$, $|2\times2|_3$ therefore $_2^{-1} = 2$
$|84|_5 = 4$, $|(4 \times 4)|_5 = 1$, therefore $_3^{-1} = 4$
$|60|_7 = 4$, $|(4 \times 2)|_7 = 1$, therefore $_4^{-1} = 2$
Then using equation (1), can be obtained as
$= |105|\ (1 \times 1)\ |_4 |_{420} + |140|\ (2 \times 1)|_3 |_{420} + |84|\ (4 \times 0)\ |_5 |_{420} + |60|(2 \times 4)|_7 |_{420} = |105 + 280 + 0 + 60|_{420}$
$= |445|_{420} = 25$

Example 1.4: This illustrates how a single error in the received residue digits is detected and corrected by employing two redundant moduli. Consider two redundant moduli, namely 13 and 17 in addition to the original moduli (4, 3, 5, 7) which gives the new moduli set (4, 3, 5, 7, 13, and 17). Now, consider the ASCII integer message with representation = 25, then the respective residue digits are (1, 1, 0, 4, 12, 8). Assume that the $_3$ digit is in error (i.e. {1, 1, 2', 4, 12, 8}). According to CRT, the integer X in the range (0, 210) can be recovered by invoking any four moduli and their corresponding residue digits, if no errors occurred in the received RNS representation.

The integer X represented as (1, 1, 2', 4, 12, 8) is recovered by considering all possible cases. Once all the possible combinations of four out of six residue digits are retained, it results in:

$(_{1,2,3,4}) = (1, 1, 2' 4) -_{1234} = 277$
$(_{1,2,3,5}) = (1, 1, 2', 12) - X_{1235} = 547$
$(r_1, r_2, r_3, r_6) = (1, 1, 2', 8) - X_{1236} = 337$
$(r_1, r_2, r_4, r_5) = (1, 1, 4, 12) - X_{1245} = 25$
$(r_1, r_2, r_4, r_6) = (1, 1, 4, 8) - X_{1246} = 25$
$(r_1, r_3, r_4, r_5) = (1, 2', 4, 12) - X_{1345} = 417$
$(r_1, r_3, r_4, r_6) = (1, 2', 4, 8) - X_{1346} = 7350$
$(r_1, r_4, r_5, r_6) = (1, 4, 12, 8) - X_{1456} = 25$
$(r_2, r_3, r_4, r_5) = (1, 2', 4, 12) - X_{2345} = 1327$
$(r_2, r_3, r_4, r_6) = (1, 2', 4, 8) - X_{2346} = 5945$
$(r_2, r_4, r_5, r_6) = (1, 4, 12, 8) - X_{2456} = 25$
$(r_3, r_4, r_5, r_6) = (2', 4, 12, 8) - X_{3456} = 1782$

Where X_{ijkl} represents the recovered result by using moduli $\{m_i, m_j, m_k, m_l\}$ as well as their corresponding residue digits $\{x_i, x_j, x_k, x_l\}$.
Notice from these results that X_{1235}, X_{1346}, X_{2345}, X_{2346}, and X_{3456} are all illegitimate numbers, since their values are out of the legitimate range [0, 419].

In the remaining seven cases, all the results are same and equal to 25, except for X_{1234}, X_{1236}, and X_{1345}. Moreover, all these results were recovered from four moduli without including m_3, that is from X_{1245}, X_{1246}, X_{1456}, and X_{2456} which are equal to 25. Hence, we can conclude that the correct result is 25 and that there was an error in r_3, which can be corrected by computing
$r_3 = |25|_5 = 0$.

Results and Discussions

The proposed scheme consists of a modified encoder and decoder pair with enhanced compression ratio, speed, security, and error detection and correction capability. Residue Number System (RNS) is applied to the ASCII character with decimal representation X which is used in the compression and encryption process using the LZW-RNS four moduli scheme. The encoder and decoder pair is then modified to work for only even n numbers for enhanced security and to detect and correct errors using RRNS.

The Proposed Encoder and Decoder

The initial dictionary with single character codes in decimals is created and then converted into its residues in a process termed as forward conversion. The encoding process is continued with the modified algorithm in residues. The compressed or encoded message is then transmitted in four bit stream channels or residuals in a particular secret order.

The secret order transmitted four bit stream channel message is received, reorganized by the decoder pair and then converted to its decimal representations through a process known as reverse conversion. The modified LZW decoding process continues until the original message is acquired back.

A good and efficient data compression algorithm is measured based on its compression rate, speed of execution, security, error detection and correction capability as well as its simplicity to implement (Kinsner and Greenfield, 1991). The performance analysis of the LZW and RNS-LZW proposed scheme is assessed based on these criteria.

Implementation of the Proposed LZW-RNS Error Handling Scheme

The error detection and correction performance analysis is done using MatLab. Given the non-redundant moduli set $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$ and redundant moduli $\{2^{2n} - 3, 2^{2n} + 1\}$ resulting in the moduli set $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1, 2^{2n} - 3, 2^{2n} + 1\}$ for the purposes of error detection and correction.

Example 1.5: This is an illustrative example to demonstrate how a single error is detected and corrected. Consider the word "**Encoding**" to be encoded and decoded using the LZW-RNS scheme, and then detect and correct errors using RRNS as follows;

For $n = 2$, the respective moduli set is {3, 4, 5, 7, 13, 17}, resulting in respective residues for the word "Encoding";

res1 = ([0, 2, 0, 0, 1, 0, 2, 1]);
res2 = ([1, 2, 3, 3, 0, 1, 2, 4]);
res3 = ([4, 0, 4, 1, 0, 0, 0, 3]);
res4 = ([6, 5, 1, 6, 2, 0, 5, 5]);
res5 = ([4, 6, 8, 7, 9, 1, 6, 12]);
res6 = ([1, 8, 14, 9, 15, 3, 8, 1]);

Any possible six out of four combinations or permutations of encoding and decoding using the LZW-RNS scheme results in the table below;

Table 1.1: Encoded and Decoded Results for Error Detection and Correction

Iteration(s)	Word to Encode, Decode and Check Errors								Decoded Word
	E	n	c	o	d	i	n	g	
1234	69	110	99	111	100	105	110	208	EncodinĐ
1235	69	110	99	111	100	105	110	688	Encodin→
1236	69	110	99	111	100	105	110	868	Encodin→
1245	69	110	99	111	100	105	110	376	EncodinŸ
1246	69	110	99	111	100	105	110	460	Encodin→
1256	69	110	99	111	100	105	110	2092	Encodin→
1345	69	110	99	111	100	105	110	103	Encoding
1346	69	110	99	111	100	105	110	103	Encoding
1356	69	110	99	111	100	105	110	103	Encoding
2345	69	110	99	111	100	105	110	1468	Encodin→
2346	69	110	99	111	100	105	110	1888	Encodin→
2456	69	110	99	111	100	105	110	4744	Encodin→
3456	69	110	99	111	100	105	110	103	Encoding

From Table 1.1, notice that 103 corresponding to possible combinations ({1, 3, 4, 5}, {1, 3, 4, 6}, {1, 3, 5, 6}, {3, 4, 5, 6}) appears most, within the [0, 210) legitimate dynamic range, and 256 ASCII character range.

Then, the missing moduli in the combination giving the most appearing message corresponding to position two (2) is modulo four ($|4|$). Hence, the residue of $|103|_4$=3, is computed and the residue {3} substituted into the last residue position of "res2" as ([1, 2, 3, 3, 0, 1, 2, 3]) instead of ([1, 2, 3, 3, 0, 1, 2, 4]). The message can thus be decoded correctly using the combination |1,2,3,4| without the redundant moduli. Finally, the correct recovered word is "**Encoding**".

Performance Analysis of the Proposed Scheme

The purpose of data compression is to reduce data size, enhance speed of transmission, and security. The gain in this research in terms of security and efficiency are enormous. Amongst others, one of the inherent properties of RNS is the reduced magnitude of computation involving the use of residues that speeds up compression. Total assess to entire network is required before hacking can be done because of the secrete order bit stream compartmental transmission.

Conclusion

The moduli set $\{2^n + 1, 2^n, 2^n - 1, 2^{n+1} - 1\}$ has been carefully chosen to assess the impact of moduli choice on the performance of the LZW algorithm, and to enhance security and encryption by constraining both the encoder and decoder pair to work for even *n* numbers. Encoder and Decoder pair has been designed using RNS for compression and encryption. The output of the LZW algorithm has also been modified to allow for secret order bit stream residual storage or transmission of data which enhances the security as well as bits requirement/reduction for transmission. It is also high in security because of the larger channel that will have to be hacked. Redundant moduli has been added for error detection and correction.

Finally, RNS has been successfully applied to the LZW algorithm leading to new LZW-RNS data compression and encryption scheme that is efficient in terms of execution speed, bits per compression, and security. Fault-tolerant schemes using RRNS which involves addition of extra moduli for the purposes of error detection and correction for either data in storage or transit has also been proposed and presented which shows better performance than the traditional LZW algorithm and other known state of the art related schemes.

Future Research Work

The proposed system is efficient in terms of security, compression efficiency, and speed of execution, as well as fault tolerance than the traditional LZW compression algorithm. Further research is to ensure multiple error detection and correction, hardware implementation, as well as the implementation of other number systems in data compression.

References

Alhassan, A., Saeed, I., & Agbedemnab, P. A. (2015). The huffman's method of secured data encoding and error correction using residue number system (RNS). *Communication on Applied Electronics (CAE) Journal, Foundation of Computer Science (FCS)*. 2 (9), ISSN: 2394-4714. Pp 14-18.

Alhassan, A. Gbolagade, K. A., & Bankas, E. K. (2015). A Novel and efficient lzw-rns scheme for enhanced information compression and security. *International Journal of Advanced Research in Computer Engineering and Technology (IJARCET), 4* (11), Pp 4015-4019.

Amit, J. (2003). Comparative study of dictionary based compression algorithms on text data. *International Journal of Computer Engineering and Applications,* I, (2), Pp.1-11.

Amusa, K. A. & Nwoye, E.O. (2012). Novel Algorithm for Decoding Redundant Residue Number System (RRNS) Codes. *IJRRAS,* Vol. 12(1), Pp 158-163.

Bankas, E. K., & Gbolagade, K. A. (2012). A speed efficient rns to binary converter for the moduli set $\{2^n, 2^n+1, 2^n-1\}$ *Journal of Computing*, *4* (5), pp. 83–88

Gbolagade, K. A., Voicu, G. R. & Cotofana, S. D. (2010). Memoryless RNS-to-binary converters, *CE Lab., Delft University of Technology*. The Netherlands, Pp 1-4.

Jane, H. Trivedi, J. (2014). A survey on different compression techniques algorithm for data compression. *International Journal of Advanced Research in Computer Science and Technology*, 2 (3), Pp1-5.

Kaur, S. & Verma, S. (2012). Design and Implementation of lzw data compression algorithm. *International Journal of Information Sciences and Techniques (IJIST)*, 2 (4), Pp 71-81.

Lu, M. (2004). *Arithmetic and logic in computer system*. John Wily and Sons, Inc, Hoboken, New Jersey.

Mahoney, M. (2012). Data compression explained. Dell Inc. http://mattmahoney.net/dc/dce.html. Access date: 11th April, 2012.

Mahyar, H. (2012). Reliable and high speed Kasumi block cipher by residue number system code. World Applied Sciences Journal. 17 (9), Pp 1149-1158.

Massachusetts Institute of Technology (MIT), (2014). Open Course Ware; Lecture, LZW, http://ocw.mit.edu/courses. Access date: 18th October, 2014.

Mohan, P. V. A. & Preemkumar, A. B. (2007). RNS-to-binary converter for four-Moduli set $\{2^n + 1, 2^n, 2^{n+1} - 1, 2^{n+1} - 1\}$. *IEEE Transactions on Circuits Systems, Reg. Papers,* 54, pp. 1245–1254.

Omondi, A. & Pumkumar, B (2007). *Residue number systems: theory and implementation.* Imperial College Press, 57 Shelton street, Covent Garden, London WC2H 9HE.

Pahami, B. (2000). *Computer arithmetic algorithms and hardware design*. Department of Electrical and Computer Engineering, University of California, Santa Barabara, Oxford University Press, New York.

Rodeh, M., Pratt, V. R., & Even. S. (1981*). A linear algorithm for data compression via string matching*. *J. ACM*, 28(1), Pp 337-343.

Shammugasundaram, S. & Lourdusamy, R. (2011). a comparative study of text compression algorithm. *International Journal of Wisdom Based Computing,* 1(3), India, Pp 68-76.

Welch, T. A. (1984). A Technique for high performance data compression. *IEEE, Sperry, Research Centre*, pp. 8-19.

Yang, L. & Hanzo, L. (2001). Redundant residue number system based error correction codes. *Vehicular Technology Conference*, VTC 2001 Fall, 3, Pp 1472 -1476.

Ziv, J. & Lempel, A. (1977). A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3).

Ziv. J & Lempel, A. (1978). Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory* 24 (5), pp. 530-536.