International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2015): 78.96 | Impact Factor (2015): 6.391

Algorithm Design to Enhance Error Detection and Correction in the ISBN

Waweru Kamaku

Abstract: ***The Conventional ISBN-13 has major limitations in Error detection and Correction capabilities with previous research showing that these capabilities could also limit the number of codewords that the code generates. This paper discusses the algorithm design of a new ISBN-16 code working modulo 17. The code is specifically designed to maximize the number of codewords and at the same time enhance error detection and correction capabilities. It is further shown that the new code addresses the weaknesses in the conventional ISBN-13 and can therefore replace it in identification of books universally.***

Keywords: ISBN, Error Detection, Error Correction

1. Introduction

The International standard book number (ISBN-13) is a unique number that identify all books. It is used as an identifier for the book for identification of the title or author (in libraries catalogues or in book's search). Due to the need for more codewords, the ISBN-13 was implemented in 2007 as an upgrade to the ISBN-10 which had historically been in use (Viklund, 2007; Eric, 2010). Kamaku etal (2012) discussed the weaknesses of the ISBN-10 and showed that it is not only limited in its dictionary (number of codewords) but also had weaknesses in error correction and detection capabilities. Kamaku etal (2012) and Kamaku(2017) analyzed the ISBN-13 and discussed showed that even if it improved on the dictionary compared to ISBN-10, the new code is weaker and limited in error detection and correction capabilities.

Communication channels always experience error related to noise, electromagnetism and even human error which means that communication can rarely guarantee an error free channel. There is therefore a need to design an ISBN code that does not only guarantee the dictionary needs but also addresses the error detection correction capabilities. This paper discusses the algorithm design for an ISBN-16 as an improvement on the ISBN-13.

ISBN-16 Algorithm Design

This section discusses the algorithm design for the ISBN-16code aimed at improving the conventional ISBN-10, ISBN-13 in error detection, error correction and overall dictionary size. The newISBN-16code is made up of all codewords;length sixteen (16) bit strings,consisting of any of the numbers 0, 1, 2, …, 9, A, B, C, D, E, F, G where A, B, C, D, E, F, G represent 10,11,12,13,14, 15and 16respectively. The choice of the letters is done to avoid confusion that may occur when the two digit numbers namely are used. For example, if one may confuse between 11 and 1,1.

Suppose $\boldsymbol{a}=a_1a_2a_3\ldots\ a_{15}a_{16}$is a ISBN-16 codeword; it must satisfy the condition

$$\sum_{i=1}^{16} ia_i \equiv 0\ (mod\ 17)\ for i = 1,2,\ldots,16 \quad \ldots\text{Equation 1}$$

Equation 1 is called a parity-check equation. All the codewords in the code are in three blocks as :

$$a_1a_2 - a_3a_4a_5a_6a_7a_8a_9a_{10}a_{11}a_{12}a_{13}\cdots a_{14}a_{15} - a_{16}$$

Generation of an ISBN-16 Codeword

a) The first block of bit strings (made up of two bit strings) is chosen randomly and represent the country where the book was published. It is estimated that there are about 194-239 countries in the world. Since two bit strings permute (out of 16 choices), there are $16^2 = 256$ possible permutations. This means that all the estimated countries will be represented.
b) The second block of digits (made up of thirteen bit strings) written in an increasing or decreasing order or constant flow; it represents the number assigned to the book by the publishing company. The flow depends on the flow of the first block of bit strings.
c) The check bit strings is chosen such that Equation 1above is satisfied.
d) Repetition of numbers is allowed.
e) The digits do not have to follow one another consecutively. That is, if you start with a 1 for an increasing order then 2 is not necessarily the next digit; it can be any of the other numbers including 1 itself since repetition is allowed.
f) To change the order of flow of bit strings from increasing to decreasing and vice versa is done by the use of a zero (0). The new flow may start with zero or any other digit. Zero (0) does not have to necessarily change the order of flow it may be used as a part of the codeword itself. That is, a zero may be used and the order of flow does not change. Here the zero acts as a neutral element. This means that the order of flow of digits does not necessarily change once a zero is put. It is at the discretion of the codeword developer to decide if to change the order or not.
g) The check digit does not necessarily to obey the order of flow of the bit strings. As seen earlier in Equation 1, the check digit obeys the given condition. It is therefore computed not chosen.
h) $(\mathbb{Z}^*_{17}, \times, +)$ is a field.

Calculation of inverses in ISBN-16Code :

If a and b are elements in the field $\mathbb{Z}$, a is said to be the additive inverse of b (denoted by – b) if

b + a $\equiv$ 0 (mod17) since 0 is the additive identity. The following basic result hold.

a	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G
-a	G	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1

Licensed Under Creative Commons Attribution CC BY

If a and b are non zero elements in the field$\mathbb{Z}$, a is said to be the multiplicative inverse of b (denoted by b^{-1}) if $a \times b \equiv 1(\text{mod}17)$ since 1 is the multiplicative identity. The following result hold.

a	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G
a^{-1}	1	9	6	D	7	3	5	F	2	C	E	A	4	B	8	G

The check digit in ISBN-16 Code

Let A= $a_1a_2a_3a_4 \cdots a_{14}a_{15}a_{16}$ be a codeword. To calculate the check digita_{16}, evaluate

P = $\sum_{i=1}^{15} ia_i \equiv 0 \text{ (mod 17)}$for $i = 1, 2, \ldots, 15$. This yield to $P + 16\, a_{16} \equiv 0\ (mod 17)$, thus

$16\, a_{16} \equiv (-P)\ (mod 17)$.Hence$a_{16} \equiv 16^{-1}(-P)\ (mod 17)$

Example 3.0.3.1

Find the check digit* in the codeword 111111111111111*

Solution: P = (1×1) + (2×1) + (3×1) + (4×1) + (5×1) + (6×1) + (7×1) + (8×1) + (9×1)+(10×1) + (11×1) + (12×1) + (13×1) + (14×1) + (15×1)= 120≡ 1 (mod17)

P + 16 a_{16}≡0 (mod17) or simply 16 a_{16}= (-P) (mod17). This yield to

a_{16}= 16^{-1} (-1) (mod17) = 16×16 (mod17) =1(mod17). The check digit is therefore 1 and the codeword is 1111111111111111.

Other examples of ISBN-16codewords include

0123456789 ABCDE3	ABCDE0EDCB A0GGGF	8105678899 90000G	6890D74222 20111D

Error Detection in ISBN-16 Code

Let A= $a_1a_2a_3a_4 \cdots a_{14}a_{15}a_{16}$ be a codeword and suppose an error(s) exist in the codeword.

To detect the error(s) the following steps are followed.

a) Count the number of bit strings in the codeword; they must be 16. If not, an error(s) exists.
b) Check if the first block of bit strings indicates the correct country where the book was published. If not then we have error(s). Since the countries are many, it is very tedious to keep checking to the respective code for each country hence unless one is sure of the code this step may be avoided. In this research, the first block of bit strings shall always be assumed to be correct since assignment of codes to each country has not been done.
c) Check the flow of the bit strings in the codeword. The flow may be either increasing or decreasing or constant.If any bit string(s) breaks the flow then we have error(s).
d) Work out for Equation 1. Even if steps a, b and c are correct, this step must be computed. It is the basis of the parity check.
e) If the steps a b, c and d are correct, then the codeword is a valid ISNB-16codeword. If any of the steps is not correct, then error(s) exist.

Example 3.1.1

Consider the codeword 810567889290000G. In this codeword the flow of bit strings is incorrect at tenth bit string. By observation, only 9 can occupy this bit string without breaking the order of flow. Replacing with 9 in this position we find 810567889290000G. Working out the parity check equation yields to

(1×8) + (2×1) + (3×0) + (4×5) + (5×6) + (6×7) + (7×8) + (8×8) + (9×9) +(10×9) + (11×9) + (12×0) + (13×0) + (14×0) + (15×0) + (16×16) ≡0 (mod 17). Hence 810567889990000G is the correct codeword.

Error Correction in ISBN-16 Code

To correct an error(s), it must first be detected. This means that the bit string(s) in error have been identified and (or)Equation 1is not met. This section deals with errors that break the order of flow or errors involving the check digit. Errors that do not break the order of flow are discussed in chapter four later.

Errors Correction on first block of bit strings

If an error exists on the first block, that is, the first block of bit strings differs from the country of publication (upon assigning each country with a specific code), then to correct the error correction is done as follows:

a) Identification of the correct country code where the book was published is done.
b) Replacement of the faulty digit with the correct one is done.
c) Identification of any other error(s) is done by through the other error(s) detection steps listed in part 3.4 above. Iferror(s) still exist, the bit string(s) in error is identified and correction is done as follows.

Check digit's Errors Correction

If the first block of bit strings digit and the order of flow of digits are correct but Equation 1is not met then error exists one the check digit (single error). To correct this:

a) $\sum_{i=1}^{15} ia_i$ (mod 17) is worked out
b) $a_{16}$is chosen such that 16a_{16} + $\sum_{i=1}^{15} ia_i$= 0 (mod17)

Example 3.2.2.1

Consider the codeword 5555566666777778. The order of flow is correct. But $\sum_{i=1}^{16} ia_i \equiv 14$ (mod 17) hence the codeword is in error. Working out the correct check digit as given in part 3.0.3 above, $a_{16} \equiv 16^{-1}$ (-P) (mod17). But P = $\sum_{i=1}^{15} ia_i \equiv 5$ (mod 17) so

$a_{16} \equiv 16^{-1}$ (-5) ≡ 16(12)≡ 5(mod17). The correct codeword is thus 5555566666777778.

Single Error correction

A single error occurs mainly due to typing mistakes or due to smudge. Suppose the order of flow is incorrect at one bit string, then a single error is said to exist if upon correcting the faulty bit string, the parity check equation holds, otherwise multiple errors exist.

Once the error has been detected and error position noted as in part 3.4 above, correction is done as follows.

a) The order of flow of bit strings (increasing, decreasing or constant) just before and after the faulty bit string is checked.
b) A number that obeys the order flow is chosen (they may be many).
c) Work out Equation 1 for each of the numbers. Once a correct one is found, replacement is done with the faulty one.

Licensed Under Creative Commons Attribution CC BY

Example 3.2.3.1 Consider the codeword 235AA053710GDBBC. We have an increasing flow of digits up to the 5th bit string, then at position six we have a zero followed by a decreasing flow but at the 9th position, order of flow is broken (error exists).
The only digits that can obey the order of flow are 3, 2, 1 and 0.

Replacing 3 with the faulty digit yields 235AA053310GDBBC; working out the parity check yields $\sum_{i=1}^{16} ia_i \equiv 10 \neq 0 \pmod{17}$ hence 235AA053310GDBBC cannot be the correct bit string.Replacing 2 with the faulty digit yields 235AA053210GDBBC; working out the parity check yields $\sum_{i=1}^{16} ia_i \equiv 0 \pmod{17}$ hence 235AA053210GDBBC is the correct codeword.

Once the correct bit string is found, there is no need to work out for other "possible" bit strings since inverses in $(\mathbb{Z}^*_{17}, \times, +)$ are unique, no two bit strings can be replaced for one faulty position to satisfy Equation 1.

2. Double error correction

Double errors occur when two bit strings are in error. The errors may be either on:

- **Any two bit strings excluding the check digit.**

This error is noticed easily since it simply occurs when any two bit strings (excluding the check digits) break the order of flow of digits. To correct these errors, the order of flow on the preceding and consequent digits is established (either increasing, decreasing or constant). A pair is chosen to replace the faulty digits such that the flow of digits is correct and the Equation 1 is obeyed.

Example 3.2.4.1.1
Consider the received codeword 810567184990000G

At position seven, the bit string 1 is between a bit string 7 and 8, breaking the order of flow leading to an error. Similarly at position nine, the bit string 4 is between a bit string 8 and 9, breaking the order of flow leading to an error. At position seven only digits 0, 7 or 8 can occupy this position without breaking the order of flow. At position nine, only 0, 8 or 9 can occupy this position without breaking the flow. Any pair that satisfies the parity check equation yields the correct codeword. Replacing the seventh and the ninth bit strings with 8 yields 810567889990000G which obeys the parity check equation hence the sent correct codeword. This choice can easily be done through trial and error though a simple computer program can be designed. Due to uniqueness of the check digit which comes as a result of unique inverses in $\mathbb{Z}_{17}$, no other pair can satisfy the equation.

- **The check digit and any other bit string**

This error occurs when one bit string and the check bit string are in error. The bit string breaks the order of flow and even when corrected, the Equation 1 is not obeyed. To correct these errors, the order of flow for the preceding and consequent digit is established (either increasing, decreasing or constant). Replacement for the faulty bit string is done and the check digit is chosen such that Equation 1 is met. Since upon replacement, different digits may obey the order of flow,the error correction may yield many different but valid codewords. This simply means that the original intended codeword must be among the possible choices and to identify it, one may need to know intended check digit on the sent codeword.

Example 3.2.4.2.1
Consider the received codeword 8105678895900005

At position ten, the bit string 5 is in between two 9's hence breaking the order of flow leading to an error. Only 0 or 9 can occupy this position without breaking the flow of digits. But upon replacing them in the position, none of the resultant codewords, 8105678890900005 or 8105678899900005, satisfy the Equation 1 hence the check digit is also in error.

Considering 8105678890900005 and using the check digit error correction method as in 3.5.2above,Bis the correct check digit hence 810567889090000B is the correct codeword.

Similarly considering 8105678899900005, G is the correct check digit hence 810567889990000G is the correct codeword.The codeword sent could therefore be any of the two codewords. No other possible codeword could have been the sent one since no other codeword can satisfy the flow of digits.

3. Triple Error Correction

Triple errors occur when three bit strings are in error. Just as in section 3.5.4 above on double errors, triple errors may occur either on any three bit strings (they do not obey the order of flow) or any two bitstring and the check bit string. The error correction is similar as in 3.5.4 above. If the error is on any three bit strings, a triple is chosen such that it obeys the order of flow and satisfy's the parity check equation. If the error is on any two bit string and the check bit string, a pair is chosen and check digit worked out such that the order of flow is obeyed and the parity check equation obeyed. Just as in 3.5.4.2 above, for errors involving the check digit, different digits may obey the order of flow. The error correction may yield many different but valid codewords. This simply means that the original intended codeword must be among the possible choices and to identify it, one may need to know intended check digit on the sent codeword.

Example 3.2.5.1 Consider the received codeword 5515566966772775
Bit string 1 at position three breaks the order of flow since from position one to seven we have an increasing flow. Similarly bit string 9 and 2 at positions eight and thirteen respectively break the order of flow. The possible bit strings that can fill these faulty (third, eighth and thirteenth) positions without breaking the order of flow are:

	Position Three	Position Eight	Position Thirteen
Case 1	0	0	0
Case 2	0	6	0

Licensed Under Creative Commons Attribution CC BY

Case 3	0	0	7
Case 4	0	6	7
Case 5	5	6	7
Case 6	5	0	0
Case 7	5	6	0
Case 8	5	0	7

Considering the non faulty positions in the codeword 55-_5566_6677_775, and taking the sum
$\sum_{i=1}^{16} ia_i (mod\ 17), i \neq 3, i \neq 8\ and i \neq 13$yields
(5× 1) +(5× 2) + (5 × 4)+(5× 5)+(6× 6)+(6× 7)+(6× 9)+ (6× 10) + (7× 11)+(7×12)+(7× 14)+(7× 15) + (5× 16) ≡16(mod 17). To get the correct codeword, the sum$\sum_{i=1}^{16} ia_i \equiv 0 (mod\ 17)$ as in Equation 1 so the faulty digits should yield 1(mod17) so that the sum
16 +1 ≡0(mod17).

Working out case by case yields the following
Case 1: (0 × 3) + (0× 8) + (0× 13) ≡0(mod17) thus incorrect
Case 2: (0 × 3) + (6× 8) + (0× 13) ≡14(mod17) thus incorrect
Case 3: (0 × 3) + (0× 8) + (7× 13) ≡6(mod17) thus incorrect
Case 4: (0 × 3) + (6× 8) + (7× 13) ≡3(mod17) thus incorrect
Case 5: (5 × 3) + (6× 8) + (7× 13) ≡1(mod17) thus correct
Case 6: (5 × 3) + (0× 8) + (0× 13) ≡15(mod17) thus incorrect
Case 7: (5 × 3) + (6× 8) + (0× 13) ≡12(mod17) thus incorrect
Case 8: (5 × 3) + (0× 8) + (7× 13) ≡4(mod17) thus incorrect
The correct codeword is 5555566666777775
Each of the above cases would yield to a valid codeword if one would work out their check digits separately.

Other Multiple Errors Correction
Errors occurring on more than 3 bit strings can also be corrected. After detecting the faulty digits positions, replacement is done on these positions such that the order of flow and the parity equation are obeyed.

Example 3.2.6.1 Consider the received codeword 117118116117111
Considering the order of flow of bit strings, the third, sixth, ninth and twelfths bit strings are in error since each of them is in between two 1's breaking the order of flow.

	Position Three	Position Six	Position Nine	Position Twelve
Case 1	0	0	0	0
Case 2	0	0	0	1
Case 3	0	0	1	0
Case 4	0	1	0	0
Case 5	1	0	0	0
Case 6	0	0	1	1
Case 7	0	1	0	1
Case 8	0	1	1	0
Case 9	1	0	0	1
Case 10	1	0	1	0
Case 11	1	1	0	0
Case 12	0	1	1	1
Case 13	1	0	1	1
Case 14	1	1	0	1
Case 15	1	1	1	0
Case 16	1	1	1	1

These sixteen cases will yield to a valid codeword if one would work out their check digits separately. For this example, the check digit is 1; working out yields

$\sum_{i=1}^{16} ia_i (mod\ 17), i \neq 3, i \neq 6,\ i \neq 9\ and i \neq 12$ yield 4(mod17). Working out case by case yields case 16 as the correct one since (1 × 3) + (1× 6) + (1× 9) + (1× 12) ≡13(mod17). But 13 + 4 ≡ 0(mod17) hence correct. The correct codeword is thus 1111111111111111.

4. Conclusion

This paper discussed the algorithm design for new ISBN-16 and analyzed its error detection and correction capabilities.It is shown that the new code by far surpasses the conventional ISBN-13 in both dictionary and error detection and correction.

References

[1] Kamaku, P. W., Kivunge, B., & Wangeci, C. (2012). ON SOME PROPERTIES AND LIMITATIONS IN THE ISBN-13 CODE. *International Electronic Journal of Pure and Applied Mathematics – IEJPAM ,4*(3), 159–165.
[2] Kamaku, W. (2017). On the Weaknesses in Error Detection and Correction in the ISBN-13, *6*(6), 2093–2096. http://doi.org/10.21275/ART20174839
[3] Kamaku, W., Mwathi, C., & Kivunge, B. (2012). LIMITATIONS IN THE CONVECTIONAL ISBN-10 CODE. *American International Journal of Contemorary Research*, *2*(2), 150–152.

Licensed Under Creative Commons Attribution CC BY