Int. J. Pure Appl. Sci. Technol., 30(1) (2015), pp. 24-28

International Journal of Pure and Applied Sciences and Technology

ISSN 2229 - 6107

Available online at **www.ijopaasat.in**

Research Paper

Transposition Error Detection in Luhn's Algorithm

Wangeci Wachira[1,*], Kamaku Waweru[1] and Lewis Nyaga[1]

[1] Department of Pure and Applied Mathematics, Jomo Kenyatta University of Agriculture and Technology

* Corresponding author, e-mail: (lcicira@gmail.com)

(Received: 12-7-15; Accepted: 27-8-15)

Abstract: *In 1950's Han's Peter Luhn formulated the Luhn formula (algorithm) to solve information storage and retrieval problem. This formula has been given worldwide application such as in the credit card numbers where it uniquely identifies an account to its respective owner among other applications. Insecurity and fraud cases associated with the credit cards have necessitated the need to analyze the algorithm error detection and correction capabilities. It is known that the transposition of digits 09 for 90 cannot be detected. The paper analyzes the detection of this transposition error and discusses key considerations in algorithm development to avoid such an error.*

Keywords: Error detection, Error Correction, Transposition error, Luhn algorithm.

1. Introduction

An error is a deviation from accuracy or correctness which is usually caused by noisy communication channels such as thermal noise, human disturbance among others. If a code word $u = 011$ is sent to the receiver through a transmitter and a code word $v = 010$ is received instead. Note that $u \neq v$ hence presence of an error in the last position. The identification of error (s) in a code word is called error detection of which the code word may be discarded or sought for retransmission. Error correction refers to the detection of errors in a code and then reconstruction of the original error free data is done. Modular Arithmetic involves working with the remainders generated by division. Let a, b, c, d be integers, a is said to be congruent to b modulo c, that is $a \equiv b \ (mod\ c)$ if $c|a - b$ or $a - b = dc$ meaning c divides the difference of a and b. (Raymond, 1986) In 2013, Khalid et al. defined Luhn formula as a formula that uses modulo 10 for error detection that is why it is sometimes referred to as the “Modulo 10” formula. The formula is used in validation of numbers against the included check digit or in calculation of a check digit. A check digit is a form of redundancy check usually at the end of a code word that can be either an alphabet or a digit used for the purpose of error detection. Checksum refers to a small sized datum computed from an arbitrary block of digital data for the purpose of error detection. (Hodge et al. 2014) The following is an illustration of how the Luhn formula works in validation of account numbers and in calculation of the check digit when required.

Validation and Check Digit Calculation Using Luhn Formula

In validation, the Luhn formula verifies a number against its included check digit. Counting from the check digit, which is the rightmost and moving left, double the value of every second digit.

If for all the products $p < 10$, then calculate $\left(\sum_{i=0}^{k} 2n_{2i} + \sum_{i=0}^{k} n_{2i+1}\right) mod\ 10$ where $k \in \mathbb{Z}$ and $n \in \mathbb{Z}_{10}$.

If there exist product $p \geq 10$, add the digits to obtain a single digit, i.e. $\forall a, b, c \in \mathbb{Z}_{10}$, $p = ab{:}\, a + b = c$ before calculation of the checksum $\left(\sum_{i=0}^{k} 2n_{2i} + \sum_{1=0}^{k} n_{2i+1}\right) mod\ 10$. If the total modulo 10 is equal to 0 (if the total ends with zero) then the number is valid according to the Luhn formula: else it is invalid.

Let "$a_1a_2 \cdots a_n$" be an account number that will have a check digit added making it of the form" $a_1a_2 \cdots a_nx$". From the rightmost double every even positioned digit to obtain "$v_1v_2 \cdots v_3x$" If $v_i \geq 10$, sum the digits of the product (e.g. 10=1+0=1, 18=1+8=9). (Ghandhi, 2010)

If for every $i, v_i < 10$ then check digit
$x = 10 - \left(checksum = \sum_{i=0}^{k} 2n_{2i} + \sum_{i=0}^{k} n_{2i+1}\right) mod\ 10$ where $k \in \mathbb{Z}$

This checksum formula only works for digits whose products with the weight is not greater or equal to 10. If there is a product whose value is greater or equal to 10 $i.e.\ \forall\, v, \exists\, v_i \geq 10$ add the digits in the of the product to obtain a single digit n then calculate the check digit by

$x = 10 - \left(checksum = \sum_{i=0}^{k} 2n_{2i} + \sum_{1=0}^{k} n_{2i+1}\right) mod\ 10$ where $k \in \mathbb{Z}$

A transposition error occurs when digits are interchanged or reversed. Adjacent transposition error occurs when two different adjacent digits are interchanged, that is if a and b are adjacent digits in a code then ab are interchanged to obtain ba. This causes change in the checksum and ultimately change in the check digit. (Gallian, 1991)

Mohr showed that Luhn formula detects all the single errors and most adjacent transpositions but not transposition of digits 09 for digits 90 since they yield the same checksum and check digit after transposition. (Mohr, 1999). According to Kamaku the strength of a code is determined by its ability to detect and correct errors. A code that does not guarantee 100% error detection gives room for manipulation and fraud which in the long run becomes detrimental to the user. A single undetected error can overshadow all the advantages associated with such a code. (Kamaku, 2012) In 1991, Gallian stated in a theorem that if an identification number $\mathrm{a_1a_2 \cdots a_n}$ satisfying $(\mathrm{a_1a_2 \cdots a_n}) \cdot (\mathrm{w_1w_2 \cdots w_n}) \equiv 0 \pmod{\mathrm{k}}$ then a single digit error $a_i \longrightarrow a_j$ is undetectable if and only if $(a_i - a_j)w_i \equiv 0(\mathrm{mod}\, k)$. Also, a transposition error in the i^{th} and j^{th} is undetectable if and only if $(a_i - a_j)(w_i - w_j) \equiv 0(mod\ k)$". Gallian (1991) proved that if an error detecting scheme with an even modulus detects all single position errors then there exist digits in the i^{th} and j^{th} which when transposed cannot be detected. Mohr (1999) states that the two most common human errors are: interchanging adjacent digits of numbers such as 37 becoming 73 and doubling the wrong one of a triple of digits, two adjacent ones of which are the same such as 557 becoming 755.

Many researchers have analyzed the various check digits algorithm but none has gone into details to explain why the transposition digits 09 for 90 pass the Luhn algorithm undetected. This paper analyses the Luhn formula in regards to the transposition error and points out the weaknesses in the formula that makes the error to pass undetected.

2. Results and Discussion:

The following theorem shows that even though $(a_i - a_j)(w_i - w_j) \neq 0\ (mod\ 10)$ (Gallian, 1991) does not guarantee 100% transposition error detection in Luhn algorithm. This will be proved by use of a counter example where digit 09 is transposed for digit 90.

Theorem 1: *Let* $a_i, a_j \in \mathbb{Z}_{10}$ *be two digits that are transposed in a code and* w_i *and* w_j *their respective weights. Then* $(a_i - a_j)(w_i - w_j) \neq a_i w_i - a_j w_i - a_i + a_j w_j$ *in Luhn algorithm.*

Proof: By counter example. If $a_i = 9, a_j = 0, w_i = 2, w_j = 1$.
$(a_i - a_j)(w_i - w_j) \neq 0\ (mod\ 10) \cdots eqn1$
$a_i w_i - a_j w_i - a_i w_j + a_j w_j = 0\ (mod\ n)$. (When $\forall\ aw \geq 10$ add the digits together before calculation of the checksum) $\cdots eqn\ 2$.

Since $eqn1 \neq eqn\ 2$ the transposition of digits 09 for 90 are undetected and vice versa.

Reducing the product to a single digit when $aw \geq 10$ makes *eqn2* to be zero, thus not equal to *eqn1*. Making the equation $(a_i - a_j)(w_i - w_j) \neq a_i w_i - a_j w_i - a_i w_j - a_j w_j$ in Luhn algorithm.

The relationship between the weights and the digits also have a high contribution in its error detection capability. The checksum difference $aw_i - bw_j$ between the two transposed digits will determine in advance if the two transposed digits will be detected as illustrated in theorem 2.

Theorem 2: *Let* $a, b \in \mathbb{Z}_{10}$ *be digits at the* i^{th} *and* j^{th} *position and* w_i *and* w_j *their respective weights. Then if Luhn detects all transposition of digits* ab *for* ba *then* $\forall a, b \in \mathbb{Z}_{10},\ 9 \nmid aw_i - bw_j$.

Proof: By contraposition, suppose $9|aw_i - bw_j$. That means $aw_i - bw_j = 9k, k \in \mathbb{Z}$. If $w_i = 2$ and $w_j = 1$. There are only two instances when $aw_i - bw_j = 9k$. That is when:

Case 1: When $a = 0$ and $b = 9$.
$aw_i - bw_j = 0 - 9 = -9 = 9(-1)$, hence a multiple of 9. From theorem 1, Luhn formula does not detect transposition of digits 09 for 90. Thus, Luhn formula does not detect all transposition of

the form ab for ba. This implies the contraposition is true, that is, Luhn formula detects all transposition of digits $ab \longrightarrow ba$ if $9 \nmid aw_i - bw_j$.

Case 2: When $a = 9$ and $b = 0$.
$bw_i - aw_j = 18 - 0 = 9 - 0 = 9(1)$, since for all $aw \geq 10$ add the digits together to obtain a single digit. Hence $aw_i - bw_j$ is a multiple of 9. From theorem 1, Luhn formula does not detect transposition of digits 09 for 90. Thus Luhn formula does not detect all transposition of digit ab for ba. This implies the contraposition is true, that is, Luhn formula detects all transposition of digit $ab \rightarrow ba$ if $9 \nmid aw_i - bw_j$.

Theorem 2 shows that every transposition of the form $ab \rightarrow ba$ and vice versa can only be detected if the difference between the product of the digits and the weights $i.e\ aw_i - bw_j$ is not a multiple of 9. It can only pass undetected if $aw_i - bw_j$ is a multiple of 9.

Proposition 1: *Let $a \in \mathbb{Z}_{10}$, $1 \leq a \leq 4$ and $w_i = 2, w_j = 1$ be weights at the i^{th} and j^{th} position respectively. Then the following holds:*

i. *Suppose an adjacent transposition of the form $a0 \longrightarrow 0a$ occurs at the i^{th} and j^{th} position, then $aw_i - aw_j = a$ in Luhn formula.*
ii. *Luhn formula detects all transposition errors of the form $a0 \rightarrow 0a$ if $aw_i - aw_j = a$.*

Proof:

i) By mathematical induction.

Basis Step: Let $a = 1 \Longrightarrow w_i - w_j = 1$, which is true since 2 - 1 = 1.
Inductive step: Assume true for $a = k$, that is $kw_i - kw_j = k$.
For $a = k + 1$,

$$(k+1)w_i - (k+1)w_j = kw_i + w_i - kw_j - w_j = k(w_i - w_j) + 1(w_i - w_j)$$
$$= k + 1$$

Hence true for $a = 1$, $a = k + 1$ whenever true for $a = 1$.

ii) From part (i), it is seen that $\forall a \in \mathbb{Z}_{10}, 1 \leq a \leq 4, aw_i - aw_j = a$ and from theorem 2, it is clearly shown that only transposition of the form $09 \rightarrow 90$ are undetected by the Luhn algorithm. Therefore, Luhn formula detects all transposition errors of the form $a0 \rightarrow 0a$ whenever $1 \leq a \leq 4$.

A code that has 100% transposition error detection has the modulus and the difference between the weights as coprimes. (Gallian, 1991) The next theorem shows that Luhn formula disagrees with this condition.

Theorem 3: *Let w_i and w_j be weights in a code modulo n. Then when the gcd $(w_i - w_j, n) = 1$ the transposition error detection in Luhn algorithm is not 100% guaranteed.*

Proof: By counter example, suppose $a_i = 9, a_j = 0, w_i = 2,\ w_j = 1$ then $w_i - w_j$=1$\Longrightarrow gcd\left(w_i - w_j, n\right) = 1$. From theorem 1, it is shown that transposition of digit 09 for 90 is undetected since $a_i w_i - a_j w_i - a_i w_j + a_j w_j = 0\ (mod\ n)$. (When $\forall\ a_i w_i \geq 10$ add the digits together). Hence $\left(w_i - w_j, n\right) = 1$ does not guarantee 100% transposition error detection for the Luhn formula.

3. Conclusion:

It is observed from the results that in Luhn formula that even though the $gcd(w_i - w_j, n)= 1$ does not guarantee 100% transposition error detection. This does not hold for Luhn algorithm since for every $a_i w_i \geq 10$ the digits of the product are added to obtain a single digit. It is observed that this move has highly contributed to the Luhn formula not been able to detect transposition errors. Probably, Luhn wanted to reduce the bulkiness in calculation of the checksum and with that came the stated detriments. Also, the difference between the products of the digits transposed, that is, $aw_i - bw_j$ highly determines if the Luhn formula will detect those transposed digit. If the difference is a multiple of 9, Luhn formula will not detect such an adjacent transposition. Even though Luhn formula is widely used, its ability to detect transposition errors is detrimental which according to Kamaku counters the benefits and strength associated with such an algorithm. (Kamaku, 2012) Therefore there is need to design an algorithm that considers all this detriments for effective error detection.

4. Recommendation

There is room for research in the analysis of the Luhn formula against the other types of errors.

References

[1] H. Raymond, A First Course in Coding Theory, Oxford University Press, New York, 1986.
[2] J. Gallian, The mathematics of identification numbers, *The College of Mathematics Journal*, 22(1991), 198-199.
[3] J. Mohr, Check Digits, (1999), Retrieved from http://www.augustana.ab.ca/~mohrj/algorithms/checkdigits.html, 23 July (2015).
[4] J. Hodge, S. Schlicker and T. Sundstrom, Algebra an Inquiry-Based Approach, CRC Press, Boca Raton, 2014.
[5] K. Ghandhi, Check digits, (2010), Retrieved from http://www.cosmos.ucdavis.edu/archives/2010/cluster6/Ghandi_Kira.pdf, 31 August (2015).
[6] W.H. Khalid, M.S. Fazlida, M. Ramlan and T.A. Mohd, Enhance Luhn algorithm for validation of credit cards numbers, *International Journal of Science and Mobile Computing*, 2(2013), 262-272.
[7] W. Kamaku, Error detection and correction on the international standard book number, *American International Journal of Contemporary Research*, 2(2012), 150-152.