

# Check Digit Schemes Based upon Weighted Multiplication and Dihedral Groups

---

Abstract Algebra – Professor Hart  
Noah Richards  
28 February 2007

## Introduction

In the end of Chapter 5, the author explores two different check digit schemes; one based upon  $D_5$ , and an extended version with  $D_{18}$ . This is known as the Verhoeff method, and has the benefit over simpler check-digit schemes in that it detects “all single-digit errors and all transposition errors involving adjacent digits”. This method was employed in 1969 by J. Verhoeff to replace the simpler methods (such as UPC check-digits) and limitations thereof.

## Methods

### UPC Method

Perhaps the simplest check-digit scheme involves a simple component-wise multiplication of the given numerals, 0-9, that catches most transposition errors and all single digit errors (note that there are no guarantees as to this). The normal UPC code is a 12-digit code, with 11-digits of data and 1-digit as a check-digit. To find the check-digit, the following method is used:

$$(a_1, a_2 a_3, \dots, a_{12}) \cdot (3, 1, 3, 1, 3, 1, 3, 1, 3, 1) \equiv 0 \pmod{10}$$

So we take, for example, the UPC on an XBOX 360© wireless controller, 8-82224-02240-8, and compute:

$$8 * 3 + 8 * 1 + 2 * 3 + 2 * 1 + 2 * 3 + 4 * 1 + 0 * 3 + 2 * 1 + 2 * 3 + 4 * 1 + 0 * 3 + 8 * 1 = 70$$

And, since  $70 \equiv 0 \pmod{10}$ , this UPC checks out fine.

For transposition errors, you can see where it would fail. For example, if we consider two adjacent digits,  $a$  and  $b$ , this scheme will fail anytime that:

$$3a + b \equiv a + 3b \pmod{10}$$

$$2a - 2b \equiv 0 \pmod{10}$$

This is true, for example, if  $a = 9$  and  $b = 4$ . In this case:

$$9 * 3 + 4 = 31 \equiv 1 \pmod{10}, \text{ and } 9 + 4 * 3 = 21 \equiv 1 \pmod{10}$$

So, this works for the pairs (9,4), (8,3), (7,2), (6,1), (5,0), and each of these with the components reversed. This can be avoided by simply never pairing any  $n$  with  $n + 5 \pmod{10}$ , but this seems unnecessarily restrictive.

Single digit changes will always cause the UPC check to fail, since  $f: Z_{10} \rightarrow Z_{10}$  where  $f$  is defined as  $f(n) = 3n \pmod{10}$  is an automorphism (since  $\gcd(3, 10) = 1$ ).

### Three-weight Scheme

The three weight scheme is nearly identical to the UPC scheme, and is used in many countries for passport numbers. The three weight scheme replaces the UPC pattern:

$$(3, 1, 3, 1, \dots, 3, 1)$$

With a pattern that repeats every three digits, such as:

$$(7, 3, 1, 7, 3, 1, \dots, 7, 3, 1)$$

This has the added benefit of catching every other digit transpositions, under the same conditions as the UPC scheme. As before, if the pairs are of the form:

$$7a + b \equiv 7b + a \pmod{10}$$

$$6a - 6b \equiv 0 \pmod{10}$$

Which fails for pairs (6, 1), (5, 0), (4, 9), (3, 8), (2, 7), and the same with the components reversed. Note that these are the exact same patterns for which the UPC scheme fails.

In addition, adjacent digit transpositions fail when:

$$7a + 3b \equiv 7b + 3a \pmod{10}$$

$$4a - 4b \equiv 0 \pmod{10}$$

So, again, this fails in the exact same way as the previous. Again, the failure patterns are:

$$2a - 2b \equiv 0 \pmod{10}$$

$$6a - 6b \equiv 0 \pmod{10}, \text{ and}$$

$$4a - 4b \equiv 0 \pmod{10}$$

In each case, dividing the left side by 2 gives:

$$a - b \equiv 0 \pmod{5}$$

$$3a - 3b \equiv 0 \pmod{5}$$

$$2a - 2b \equiv 0 \pmod{5}$$

And, when  $a - b \equiv 0 \pmod{5}$ , each equation simplifies to:

$$k(a - b) \equiv 0 \pmod{5} \rightarrow k \equiv 0 \pmod{1}$$

Which is true for all  $k$ . Thus for any check digit scheme of this form, where the difference between adjacent multipliers is an even number, digits that are  $\pm 5$  of each other will transpose and not be caught.

This may seem a fallback until you realize that this is the best case you can hope for with this type of scheme. If, instead of choosing multipliers that were relatively prime to 10, say 5 and 2, you end up with situations where single digit mistakes will not be caught. For example, with the multiplier 5, any even digit will be equivalent, or with the multiplier 2, both 0 and 5 are the same value.

This also implies that the most complicated scheme of this type would be a Four Weight scheme, since  $U(10) = \{1, 3, 7, 9\}$ , and the weights used must be elements of  $U(10)$  to be able to catch single digit transpositions.

So, perhaps the every-other digit transposition is common in these applications, and thus this makes more sense than the UPC scheme, but it seems that there is very little benefit from such a little change. Both methods suffer from the  $\pm 5$  issue.

## Verhoeff's Method

This method was based upon permutations in  $D_5$  (designed for digits 0-9, since  $|D_5| = 2 * 5 = 10$ ), designed to catch transposition errors that the modulo-based method cannot. Gallian, in his paper and in the textbook, gives the following permutation as an example:

$$\sigma = (01589427)(36)$$

For the sake of brevity, the table of this is not listed here, but can be found on page 109 of the textbook.

The mapping  $f: Z_{10} \rightarrow D_5$  is given by:

$$f(n) = \begin{cases} e, \alpha, \alpha^2, \alpha^3, \alpha^4 & 0 \leq n \leq 4 \\ \beta, \alpha\beta, \alpha^2\beta, \alpha^3\beta, \alpha^4\beta & 5 \leq n \leq 9 \end{cases}$$

Like the UPC scheme, we simply tack on a digit such that:

$$\sigma(a_1) + \sigma^2(a_2) + \dots + \sigma^n(a_n) \equiv 0 \pmod{10}$$

(The version used with German banknotes was slightly different, which used  $a_{11}$  instead of  $\sigma^{11}(a_{11})$  for the last digit)

The author shows, on page 110, a table that includes the powers of  $\sigma$  for easier computation.

As Gallian mentions in his paper, this method has been extended to handle such numerals as credit card numbers, allowing for merchants to check validity of numbers based upon the final check digit.

## Other considerations

As a computer scientist studying these methods, I find it important to study the algorithmic complexity of each. For applications such as credit card checks, if the middleware application does not verify the

number, the credit card company must verify it for them. This can become incredibly expensive for companies that have to validate millions of transactions a day.

In terms of computing bottlenecks, both algorithms are effectively the same. Each does some matter of lookup for individual digit values (either as a multiply or as a lookup-table, as would necessarily be the case for the Verhoeff method), multiplies a number of digits together, and then performs a division and a remainder operation.

The chief difference lies in the size of the lookup table. For the Verhoeff in  $D_5$ , the size of the lookup table is 100 entries (note that I am referring to the table that displays powers of  $\sigma$ , not the table that displays the simple multiplications). For the larger Verhoeff method, the lookup table (for  $D_{18}$ ) is  $36 * \text{number of digits}$ , e.g. 396 entries for a 10 digit + 1 checksum number. This is still not large, by any stretch of the imagination.

The act of multiplying through the digits and taking the remainder can be parallelized by a method similar to Google's map-reduce program (Dean, 2004). With credit card checking, for example, each group of four digits could be sent along different paths, reducing each to a number modulo 10. Then, the resulting 4 numbers can be themselves sent out to be multiplied and reduced modulo 10, checking to make sure the final digit is 0.

## Conclusions

Since the Verhoeff method is not considerably more expensive algorithmically than the weighted schemes, it seems logical to replace weighted schemes with schemes based upon Dihedral groups. The weighted schemes fail to catch a decent amount of transposition errors, both when the pair of numerals are  $\pm 5$  of each other, and when the numerals have equivalent weights (which varies, depending on the scheme). Also, the weighted schemes are limited to four weight at a maximum, and thus as the number of digits increases, you become more likely to make errors. In addition, two of the interchangeable pairs, (9,4) and (7,2), are actually visually rather close, and thus you can imagine that a transposition error involving two of these digits is quite possible.

## **Bibliography**

Dean, J. a. (2004). MapReduce: Simplified Data Processing on Large Clusters. *OSDI'04: Sixth Symposium on Operating System Design and Implementation*. San Francisco, CA.

Gallian, J. (1996). Error Detection Methods. *ACM Computing Surveys* , 28 (3), 504-514.