Error Detection Methods

Joseph A. Gallian
Department of Mathematics and Statistics
University of Minnesota, Duluth
Duluth, Minnesota 55812

June 20, 2003

The use of check digits with identification numbers for error detection is now standard practice. Noteable exceptions such as social security numbers, telephone numbers and serial numbers on currency predate computers. Despite their ubiquity and utility, few people are knowledgeable about the myriad of check digit schemes in use by businesses. In this article we survey many of these schemes. Among them are three that have not been described in journal articles previously.

Categories and Subject Descriptors: A.1 [**Introductory and Survey**]; E.4 [**Error Control Codes**]; G.2 [**Discrete Mathematics**]

General Terms: Check Digits, Error Detection, Error Correction

Additional Key Words and Phrases: Codes

1. Introduction

The availability of inexpensive, fast, reliable scanning devices and computers has made the appendage of a check digit to identification numbers a standard practice. Indeed, one finds check digits appended to identification numbers on airline tickets, credit cards, money orders, bank accounts, checking accounts, library books, grocery items, travelers checks, driver's licenses,

passports, rental cars, chemicals, automobiles, blood bank items, photofinishing envelopes, UPS packages, express mail, bar coded mail and books. The presence of a check digit enables a computer to detect errors in numbers. Dozens of detection schemes have been devised. In this article we discuss many of the methods that are being used around the world. Information about proposed methods that have not been put into practice is contained in the references. Many of the proposed methods described in the references are superior to those in use.

2. The UPC Schemes

The identification number code people most frequently encounter is the Universal Product Code found on grocery items [5, p.7-11], [40]. A UPC identification number consists of twelve digits, each of which can be from 0 to 9. The first digit identifies a broad category of product type (for example, 3 signifies a health product), the next five digits identify the manufacturer, the next five the product, the last is the check digit. (For many items, the check digit is not printed, but it is always bar coded.) The check digit a_{12} for the UPC number $a_1 a_2 \ldots a_{11}$ is chosen to satisfy the condition

$$(a_1, a_2, \ldots, a_{12}) \cdot (3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1) =$$

$$3a_1 + a_2 + 3a_3 + \ldots + 3a_{11} + a_{12} \equiv 0 \pmod{10}.$$

In Figure 1, the check digit is 5 because

$3 \cdot 0 + 3 + 3 \cdot 8 + 0 + 3 \cdot 0 + 0 + 3 \cdot 1 + 3 + 3 \cdot 7 + 1 + 3 \cdot 0 = 55$ and $55 + 5 = 60 \equiv 0 \pmod{10}$.

Figure 1. UPC identification number 03800013710 and check digit 5.

Notice that any single error, say $a_1a_2 \ldots a_i \ldots a_{12} \rightarrow a_1a_2 \ldots a_i^{'} \ldots a_{12}$, is detectable since $(a_1, a_2, \ldots, a_i^{'}, \ldots, a_{12}) \cdot (3, 1, 3, 1, \ldots, 3, 1) \not\equiv 0 \pmod{10}$ if $a_i \neq a_i^{'}$. To see this observe that if both

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

and

$$(a_1, a_2, \ldots, a_i^{'}, \ldots, a_{12}) \cdot (3, 1, 3, 1, \ldots, 3, 1) \equiv 0 \pmod{10}$$

then $((a_1, a_2, \ldots, a_i, \ldots, a_{12}) - (a_1, a_2, \ldots, a_i^{'}, \ldots, a_{12})) \cdot (3, 1, 3, 1, \ldots, 3, 1) \equiv 0 \pmod{10}$. This reduces to either $a_i - a_i^{'} \equiv 0 \pmod{10}$ or $3(a_i - a_i^{'}) \equiv 0 \pmod{10}$. But $a_i \neq a_i^{'}$ implies $a_i - a_i^{'} = \pm 1, \pm 2, \pm 3, \pm 4, \pm 5, \pm 6, \pm 7, \pm 8, \pm 9$, so we have a contradiction in both cases.

There is an abbreviated 8-digit version of the 12-digit UPC code called *Version E* (see Figure 2) that is often used in special circumstances, such as round containers (for example, soft drink cans), small items, and magazines. The Version E scheme uses four formulas depending on the last non-check digit of the number $a_1a_2a_3a_4a_5a_6a_7$ to append a check digit [40, p.8].

1. If a_7 is 0, 1 or 2, the check digit a_8 is chosen so that $3a_1 + a_2 + 3a_3 + 3a_4 + a_5 + 3a_6 + a_7 + a_8 \equiv 0 \pmod{10}$.

2. If a_7 is 3, the check digit a_8 is chosen so that $3a_1 + a_2 + 3a_3 + a_4 + a_5 + 3a_6 + a_8 \equiv 0 \pmod{10}$.

3. If a_7 is 4, the check digit a_8 is chosen so that $3a_1 + a_2 + 3a_3 + a_4 + 3a_5 + 3a_6 + a_8 \equiv 0 \pmod{10}$.

4. If a_7 is 5, 6, 7, 8 or 9, the check digit a_8 is chosen so that $3a_1 + a_2 + 3a_3 + a_4 + 3a_5 + a_6 + 3a_7 + a_8 \equiv 0 \pmod{10}$.

Figure 2. The UPC Version E code.

The apparent irregular patterns for the weights in the four cases are the result of the way the 12 digit UPC numbers are converted to 8 digit numbers. All Version E numbers correspond to 12 digit numbers that have 0s in four consecutive positions that have been suppressed [40]. To explain just the first of the four cases, consider a company whose manufacturer's number is 49100. Since this number ends with two 0s, this company must assign all products that will use a Version E code with a 5 digit number that begins with two 0s. The 12 digit number would then have the form $04910000a_9a_{10}a_{11}a_{12}$. In cases where the third digit of the manufacturer's number is 0, 1 or 2 (the initial 0 is not part of the manufacturer's number), the third digit is then moved to position 11 and the four 0s are suppressed. Thus the 12 digit number $04910000a_9a_{10}a_{11}a_{12}$ becomes $049a_9a_{10}a_{11}1a_{12}$. When the weight 1 corresponding to the third digit of the manufacturer's number is placed in position 11 and the weights corresponding to the four suppressed 0s are also suppressed we obtain the weight vector $(3, 1, 3, 3, 1, 3, 1, 1)$ used in the first formula given above.

There is also a UPC shipping container code that employs two check digits. One check digit is the standard UPC check digit and the second is calculated using modulo 103 arithmetic [41, p.5].

European countries use a 13 digit analog of the UPC number called the European Article Number (EAN) [27]. In this case the weight vector is (1,3,1,3,...,1,3,1). For EAN numbers the manufacturer's number is preceded by two digits that identifies the country in which the item was produced.

3. The Credit Card Scheme

A more complicated scheme, developed by IBM, is used by credit card companies, libraries, blood banks, photofinishing companies, pharmacies, the motor vehicle divisions of South Dakota and Saskatchewan, and some German banks [24], [15]. In this case, let σ be the permutation defined by $\sigma(0) = 0, \sigma(1) = 2, \sigma(2) = 4, \sigma(3) = 6, \sigma(4) = 8, \sigma(5) = 1, \sigma(6) = 3, \sigma(7) = 5, \sigma(8) = 7, \sigma(9) = 9$. For any string of digits $a_1a_2 \ldots a_{n-1}$ we assign the check digit a_n so that $\sigma(a_1) + a_2 +$

$\sigma(a_3) + a_4 + \cdots + \sigma(a_{n-1}) + a_n \equiv 0 \pmod{10}$. (When n is odd, σ is applied to the even numbered positions instead.) Let us look at an example. Say we have the number 7659214. Then the check digit c satisfies $5 + 6 + 1 + 9 + 4 + 1 + 8 + c \equiv 0 \pmod{10}$ so that $c = 6$.

The credit card shown in Figure 3 is reproduced from an ad promoting VISA card. Notice that the check digit on the card is not be valid since $\sigma(4)+4+\sigma(1)+7+\sigma(1)+2+\sigma(3)+4+\sigma(5)+6+\sigma(7)+8+\sigma(9)+1+\sigma(1)+2 = 8+4+2+7+2+2+6+4+1+6+5+8+9+1+2+2 = 69 \not\equiv 0 \pmod{10}$. Thus, the check digit does not " check".

Figure 3. VISA card with an invalid number.

We mention in passing that the IBM check digit method described above is the basis for a recent story in the *New York Times* [14] about a computer program known as Credit Master found on many electronic bulletin boards that produces fake credit card numbers that have the correct check digit.

Both the UPC scheme and the IBM scheme detect 100% of all single digit errors, while neither detects 100% of all errors involving the transposition of adjacent digits. In particular, an error of the form $\ldots ab \ldots \rightarrow \ldots ba \ldots$ is undetected by the UPC scheme if $|a - b| = 5$ and it is undetected by the IBM scheme if $|a - b| = 9$. Thus, of the 90 possible transposition errors the UPC scheme detects all except the pairs 05, 16, 27, 38, 49 and their reversals. In the case of the IBM scheme only the transpositions of 09 and 90 are undetected. So, the

UPC code detects 80 out of 90 or 88.9% and the IBM scheme detects 88 out of 90 or 97.8%.

The state of Wisconsin uses a 14 character driver's license number. The first character of the number is the first character of the person's last name (S for Smith, J for Jones etc.). This is followed by 10 digits that are determined by the person's last name, first name, middle initial and date of birth [22, pp.292-296]. Finally, there is a tie breaking digit to prevent the same number being assigned to different people and a check digit. To calculate the check digit, the lead letter is converted to a digit using the assignment $A \to 1, B \to 2, \ldots, I \to 9, J \to 1, K \to 2, \ldots, R \to 9, S \to 2, T \to 3, \ldots, Z \to 9$. (Notice the aberration at S.) The credit card method is then applied to the thirteen digits to obtain the check digit.

4. 3-weight Schemes

After single digit errors and errors involving the transposition of adjacent digits, the next most common errors are those of the form $\ldots abc \ldots \to \ldots cba \ldots$ (such errors are called *jump transposition*) [42]. For example, for a number such as 726-5258 one might naturally transpose "52" and "58". This error would be undetected by the UPC and IBM schemes but would be detected by the one used by American banks and by the one used on passports in many Western countries. To a number $a_1 a_2 \ldots a_8$ banks assign $(a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8) \cdot (7, 3, 9, 7, 3, 9, 7, 3) \pmod{10}$ [24]. Similarly, many countries use the weighting vector $(7, 3, 1, 7, 3, 1, \ldots)$ and modulo 10 arithmetic to assign check digits to numbers appearing on passports [13]. These 3-weight schemes detect all the errors that the UPC method does but they also detect errors of the form $\ldots abc \ldots \to \ldots cba \ldots$ as long as $|a - c| \neq 5$. (Clearly, neither the UPC scheme nor the IBM scheme detects these errors.) In particular, the 3-weight schemes detect 100% of single digit errors, 88.9% of transposition errors involving adjacent digits and 88.9% of the jump transposition errors. In the case of errors of the form $\ldots aca \ldots \to \ldots bcb \ldots$ (called *jump twin errors*) and those of the form $\ldots aa \ldots \to \ldots bb \ldots$ (called *twin errors*) the detection rates depend on the length of the number and the choice of the three weights. For example, for numbers of length 10 (including the check digit) the

weight vector (1,3,7,1,3,7,1,3,7,1) detects 66.7% of the jump twin errors whereas the weight vector (7,3,1,7,3,1,7,3,1,7) detects 55.6% of the jump twin errors. For single digit errors, transposition errors and jump transposition errors there is nothing special about the order in which the weights occur except that any three consecutive weights must be distinct and relatively prime to 10 (see Theorem 1 below). For example, it is just as well to use 1, 9, 7, 1, 9, 7, ...or 3, 9, 7, 3, 9, 7, ...and so on. Analogously, one could use four weights such as 1, 3, 7, 9, ...or 1, 3, 9, 7, ...in a modulus 10 scheme although I do not know of any instance where this has been done in practice. A 4-weight scheme detects transposition errors involving adjacent digits and jump transposition errors at the rate of 88.9%. Interestingly, with regard to jump twin errors and twin errors the order of the four weights does matter. For example, using the order 1, 3, 9, 7,... no jump twin errors are detected whereas using the order 1, 3, 7, 9,... permits the detection of 88.9% of jump twin errors. To see why consider, for instance, an error of the form $aca\ldots \rightarrow bcb\ldots$. Using the order 1, 3, 9, 7 the digits aca contribute $a \cdot 1 + c \cdot 3 + a \cdot 9 = 10a + 3c = 3c \pmod{10}$ whereas the digits bcb contribute $b \cdot 1 + c \cdot 3 + b \cdot 9 = 10b + 3c = 3c \pmod{10}$. Thus the error is not detected. A similar argument applies no matter where the jump twin error occurs. In contrast, using the weights 1, 3, 7, 9 the digits aca contribute $a \cdot 1 + c \cdot 3 + a \cdot 7 = 8a + 3c \pmod{10}$ whereas the digits bcb contribute $b \cdot 1 + c \cdot 3 + b \cdot 7 = 8b + 3c \pmod{10}$. But when $|a - b| \neq 5$, it follows that $8a \neq 8b \pmod{10}$ and the error is detected. Thus this ordering detects 88.9% of the jump twin errors.

The detection rate for twin errors depends on both the length of the number and the ordering of the weights. The weight vector (1,3,9,7,1,3,9,7,1,3) detects 88.9% of the twin errors whereas the vector (1,3,7,9,1,3,7,9,1,3) detects 49.4% of the twin errors.

There is a modified generalized IBM code described in [15] that is an improvement of the IBM method that detects 95.6% of jump transposition errors and twin errors but I do not know of any instances where it is in use.

5. Modulus 11 Schemes

In contrast to the schemes previously described, the method used to assign a check digit to the International Standard Book Number (ISBN) employs the modulus 11 [38]. In particular, the check digit a_{10} is chosen to satisfy the condition $(a_1, a_2, \ldots, a_{10}) \cdot (10, 9, 8, 7, 6, 5, 4, 3, 2, 1) \equiv 0 \pmod{11}$. This method detects 100% of all single digit errors and 100% of all transposition errors (not just those involving adjacent digits). The drawback of this method is that in some cases the check digit is required to be 10, which is not a single digit. To maintain a uniform ten character format for all books the character X is used to represent 10.

There are many variations of the ISBN scheme that are designed to avoid the introduction of the alphabetic character. Typical of these is an IBM scheme used by the states of Arkansas, New Mexico and Tennessee to assign driver's license numbers [19]. To the 7-digit number $a_1a_2a_3a_4a_5a_6a_7$ is assigned the check digit $-(a_1, a_2, a_3, a_4, a_5, a_6, a_7) \cdot (2, 7, 6, 5, 4, 3, 2) \pmod{11}$ unless this number is 0 or 1. (Recall $-3 = 8 \pmod{11}$ and so on.) In these two instances, 1 or 0 is appended respectively. This method catches all single digit errors but not all transposition errors. The errors of the form $\ldots a_i \ldots a_j \ldots \rightarrow \ldots a_j \ldots a_i \ldots$ that go undetected are those where $i = 1$ and $j = 7$ (an unlikely error indeed) and some involving the check digits 0 and 1. Of course, one could avoid the use of an alphabetic character by simply not issuing numbers that yield the dot product 10. This scheme detects 100% of single digit errors and 98.2% of transposition errors involving adjacent digits. Nothing would be lost if the weight vector began with 8 instead of 2 and there would be a slight gain since errors of the form $a_1a_2 \ldots a_7a_8 \rightarrow a_7a_2 \ldots a_1a_8$ would be detected. Table 1 gives a catalog of common "pattern" errors and their relative frequency as found in one empirical study [42]. Phonetic errors are those of the form $\ldots a0 \ldots \leftrightarrow \ldots 1a \ldots$ for $a = 2, 3, \ldots, 9$. When giving a credit card number over a telephone, for instance, "fifty" might well be interpreted as "fifteen". Obviously, phonetic errors are language dependent. The frequency for phonetic errors in Table 1 is based on English, Dutch and German. A type of error not listed in Table 1 are format errors caused by the insertion or deletion characters. These are quite common and are automatically detected when the numbers have a fixed length.

Table 1. Common Pattern Errors

Error type	Form	Relative frequency
single error	$a \to b$	79.1%
transposition of adjacent digits	$ab \to ba$	10.2%
jump transposition	$abc \to cba$	0.8%
twin error	$aa \to bb$	0.5%
phonetic error	$a0 \leftrightarrow 1a$	0.5%
	$a = 2, \ldots, 9$	
jump twin error	$aca \to bcb$	0.3%

A scheme designed to detect many kinds of errors is a modulo 11 scheme used by some German banks that employs weights that form a geometric progression rather than the arithmetic progression as in the case for the ISBN method [15]. Specifically, a_n is chosen so that $(a_1, a_2, \ldots, a_n) \cdot (2, 2^2, \ldots, 2^n) \equiv 0 \pmod{11}$. As before, the situation that $a_n = 10$ must be avoided or handled in some special way. Whereas the ISBN method detects 100% of single errors, 100% of transposition errors and 100% of jump twin errors it does not detect 100% of phonetic errors or 100% of twin errors. On the other hand, for $n \leq 10$, the weighting vector $(2, 2, \ldots, 2^n)$ permits 100% detection of all errors listed in Table 1.

Of all the methods actually in use the most exotic I have encountered is the one used on some German bank accounts [15]. This scheme, called the P.T.T. scheme, employs three permutations and two moduli as follows. For $i = 1, 2$ and 3 define $\sigma_i(a) = (i(a+1) (\mathrm{mod} 11)) \pmod{10}$. To the number $a_1 a_2 \ldots a_8$, assign

$$\sigma_1(a_1)+\sigma_2(a_2)+\sigma_3(a_3)+\sigma_1(a_4)+\sigma_2(a_5)+\sigma_3(a_6)+\sigma_1(a_7)+\sigma_2(a_8) \pmod{10}.$$

As an illustration consider 2191-06-70. Here the check digit is

$$\sigma_1(2) + \sigma_2(1) + \sigma_3(9) + \sigma_1(1) + \sigma_2(0) + \sigma_3(6) + \sigma_1(7) + \sigma_2(0) =$$

$$3 + 4 + 8 + 2 + 2 + 0 + 8 + 2 \equiv 9 \pmod{10}.$$

This scheme detects 100% of single digit errors, 96.3% of transposition errors involving adjacent digits and jump transpositions, 95.6 % of twin

and jump twin errors and 95.8% of phonetic errors.

6. Modulus 9 and 7 Schemes

Because the most common error is a single digit error, it is surprising that there are several error detection schemes in widespread use that do not detect 100% of single digit errors. Indeed, United States Postal Service money orders, VISA travelers checks, airline tickets, Federal Express mail and United Parcel Service packages are assigned check digits that use modulus 9 or modulus 7 schemes which are not 100% effective at detecting single errors. The U.S. postal money orders, for instance, simply divide the identification number by 9. The remainder is the check digit [24]. For VISA travelers checks, the check digit is the digit that must be added to the identification number so that the resulting number is evenly divisible by 9 [24]. For example, dividing 1002044679091 by 9 gives a remainder of 7 so the check digit is 2. These modulus 9 methods detect all single digit errors except substitution of a 9 for a 0 or vice versa. The only transposition errors involving adjacent digits detected by this method are those involving the check digit. It follows that single errors are detected at the rate of 98.0%. The rate for detecting errors involving the transposition of adjacent digits depends on the length of the number but in all practical situations it is low. For example, for money orders it is only 10%.

Airline companies, Federal Express and the United Parcel Service use the remainder upon division by 7 to assign check digits [24]. This method is slightly less effective than dividing by 9 for detecting single digit errors but fairly effective for detecting transposition errors. In particular, any substitution of b for a in the identification number where $|a - b| = 7$ will go undetected while any transposition of digits a and b with $|a - b| = 7$ will also go undetected. The single error detection rate for this method is 94.0% whereas the rate for detecting adjacent transposition errors is 94.1%.

7. Alphanumeric Schemes

Many identification numbers utilize both alphabetic and numeric characters. One of the most prevalent of these was developed in 1975 and is called Code 3-out-of-9 (the name derives from the method of bar coding of the characters), or simply Code 39 [27]. Code 39 permits the 26 upper case letters A through Z, the digits 0 through 9 and the characters -, . and a space. Because Code 39 has been chosen by the Department of Defense, the automotive companies and the health industry for use by their suppliers it has become the workhorse of non-retail business. A typical example of a Code 39 number is 210SA32ZBV. The last character is the "check." The check character is determined by assigning the letters A through Z the numerical values 10 through 35 respectively. The values 36, 37 and 38 are assigned to -, . and a space, respectively. The original number comprised of the digits 0 through 9, the letters A through Z and dashes, periods and spaces is now converted to a string $a_1, a_2, \ldots, a_n$ where the a_i's are integers between 0 and 38. The check character is the character corresponding to the numerical value

$$(a_1, a_2, \ldots, a_n) \cdot (n, n-1, \ldots, 2, 1) \pmod{39}.$$

Finally, this value is converted to its alphabetical counterpart, if necessary. Let us examine the Code 39 method for the number 210SA32ZBV. Here is how we determine that the check character is V. First we convert the alphabetic characters to their numeric counterparts : $210SA32ZB \rightarrow$2, 1, 0, 28, 10, 3, 2, 35, 11. Then we compute

$$9 \cdot 2 + 8 \cdot 1 + 7 \cdot 0 + 6 \cdot 28 + 5 \cdot 10 + 4 \cdot 3 + 3 \cdot 2 + 2 \cdot 35 + 1 \cdot 11$$

$$= 18 + 8 + 0 + 168 + 50 + 12 + 6 + 70 + 11 = 343.$$

Since 343 divided by 39 has a remainder of 31 and V has the numerical value 31, the check character is V. In many applications of Code 39 the special characters \$, /, + and % are permitted. These characters are assigned the numerical values 39 through 42, respectively. In these applications the check character is determined by the remainder upon division by 43 instead of 39. When the modulus 39 is used the single error detection rate depends on the number of characters in the number and is less than 100%. Precise information about which errors are detectable in contained in Section 9. For the modulus 43, 100% of the single digit errors and 100% of the transposition errors are detected.

In some cases where the 43 character version of Code 39 is used the check digit simply corresponds to the modulo 43 value of the sum of the values of the characters (i.e., weights are not used). For example, to compute the check character for the number E598976987 we calculate $14 + 5 + 9 + 8 + 9 + 7 + 6 + 9 + 8 + 7 = 82 \equiv 39 \pmod{43}$ so that the check character is \$. This method detects 100% of single digit errors but 0% of the transposition errors not involving the check digit.

The state of Washington and the province of Manitoba use a check digit scheme on their driver's license numbers [19]. The license number is a blend of 12 alphabetic and numeric characters. To compute the check digit, alphabetic characters are assigned numeric values as follows: $* \rightarrow 4, A \rightarrow 1, B \rightarrow 2, \ldots, I \rightarrow 9, J \rightarrow 1, K \rightarrow 2, \ldots, R \rightarrow 9, S \rightarrow 2, T \rightarrow 3, \ldots, Z \rightarrow 9$. (Notice the aberration at S.) The 12-character license number, after the alphabetic to numeric conversion, then corresponds to a string of digits $a_1 a_2 \ldots a_{12}$ with a_{10} as the check digit calculated as $|a_1 - a_2 + a_3 - a_4 + \cdots a_9 - a_{11} + a_{12}| \pmod{10}$. The absolute value was introduced to avoid negative numbers but since $-1 \equiv 9 \pmod{10}$ instead of using the absolute value, the weight vector $(1, 9, 1, 9, \ldots, 1)$ could have been used. Interestingly, the use of the absolute value rather than the weight vector $(1, 9, 1, 9, \ldots, 1)$ actually reduces the error detection capability of the scheme.

Although it is common practice to put the check digit last, notice that the above scheme does not do so. There is no mathematical reason why the check digit should be last. Besides the state of Washington, South Dakota does not put the check digit last on their driver's license numbers [19].

8. Error Correcting Schemes

Schemes that incorporate two check digits are not uncommon. For instance, Norway employs two check digits modulo 11 to allocate registration numbers to its citizens [28]. The last two digits of these eleven

digit numbers $a_1 a_2 \ldots a_{10} a_{11}$ are chosen so that

$$(a_1, a_2, \ldots, a_{10}) \cdot (3, 7, 6, 1, 8, 9, 4, 5, 2, 1) \equiv 0 \pmod{11} \text{ and}$$

$$(a_1, a_2, \ldots, a_{11}) \cdot (5, 4, 3, 2, 7, 6, 5, 4, 3, 2, 1) \equiv 0 \pmod{11}.$$

This method detects all single digit errors and all double errors except those where the difference between the correct number and the incorrect number has the form $(0, 0, 0, a, 0, 0, 0, 0, 0, 11-a, 0)$. Numbers for which a_{10} or a_{11} is "10" are not assigned.

An even more effective two check digit scheme consists of strings of length 10 satisfying $(a_1, a_2, \ldots, a_{10}) \cdot (1, 1, \ldots, 1, 1) \equiv 0 \pmod{11}$ and $(a_1, a_2, \ldots, a_{10}) \cdot (1, 2, 3, \ldots, 9, 10) \equiv 0 \pmod{11}$. Avoiding all strings that require a_9 or a_{10} to be "10" still leaves 82,644,629 numbers. This method detects all double errors and *corrects* all single errors. The first dot product determines the magnitude of any single error while the second one identifies the location of the error. Let's see how this works. Say our number is 73245018. Then a_9 and a_{10} satisfy $8 + a_9 + a_{10} \equiv 0 \pmod{11}$ and $10 + 9a_9 + 10a_{10} \equiv 0 \pmod{11}$ so that $a_9 = 7$ and $a_{10} = 7$. Consider the error $7324501877 \to 7824501877$. Since the sum of the digits of the incorrect number is $5 \pmod{11}$, we know that one of its digits is 5 too large (assuming only one error has been made). But which one? Suppose the error occurred in the ith position. Then the second dot product is $5i$ too large. That is, $(7, 8, 2, 4, 5, 0, 1, 8, 7, 7) \cdot (1, 2, 3, 4, 5, 6, 7, 8, 9, 10) \equiv 5i \pmod{11}$ or $10 \equiv 5i \pmod{11}$. We conclude that the second digit is 5 too large.

Single digit errors in reading bar coded identification numbers are usually correctable in much the same fashion. An unintelligible block of bars pinpoints the source of the error while the check digit condition diagnoses the extent of the error.

Two-check-digit schemes that correct all single digit errors and all transposition errors utilizing modulo 37 or modulo 97 arithmetic have been discovered ([36] and [7]). An industrial bar code scheme called Code 93 also uses two check digits [27, p.34]. One large mail order house has implemented a four-check-digit scheme [43].

9. Theory

The examples above can be put in an abstract setting as follows. Let Z_k be the set $\{0, 1, 2, \ldots, k-1\}$ and let $\sigma_1, \sigma_2, \ldots, \sigma_n$ be a sequence of mappings from Z_k into itself. For any string of elements $a_1 a_2 \ldots a_{n-1}$ from Z_k, append an element a_n so that $\sigma_1(a_1) + \sigma_2(a_2) + \ldots + \sigma_n(a_n) \equiv 0 \pmod{k}$. We call the sequence $\sigma_1, \sigma_2, \cdots, \sigma_n$ a *check digit scheme* for Z_k. Typically, σ_n is chosen to be the identity or the negative of the identity. A single digit error $a_i \rightarrow a_i'$ is detectable if and only if $\sigma_i(a_i) \not\equiv \sigma_i(a_i') \pmod{k}$ while a transposition error $\ldots a_i a_{i+1} \ldots a_j a_{j+1} \ldots \rightarrow \ldots a_j a_{i+1} \ldots a_i a_{j+1} \ldots$ is detectable if and only if $\sigma_i(a_i) + \sigma_j(a_j) \not\equiv \sigma_i(a_j) + \sigma_j(a_i) \pmod{k}$. Of course, the condition for detection of a single digit error in position i is just that σ_i is a permutation on Z_k. Since the mapping from $Z_k \rightarrow Z_k$ given by $x \rightarrow mx$ for all x is a permutation if and only if $\gcd(m, k) = 1$, we see that the UPC, the bank, passport and the ISBN schemes detect 100% of all single-digit errors, whereas a scheme that assigns a check digit a_n to the string $a_1 a_2 \ldots a_{n-1}$ so that $(a_1, a_2, \ldots, a_{n-1}, a_n) \cdot (n, \ldots, 2, 1) \equiv 0 \pmod{10}$ does not. In particular, notice that a single error $a_i \rightarrow a_i'$ for i even is undetected if $|a_i - a_i'| = 5$. Similarly, a single error $a_i \rightarrow a_i'$ for i that is divisible by 5 is undetected if $|a_i - a_i'|$ is even. Despite this deficiency, the latter method is used on the Chemical Abstract Service registry numbers [24] and on driver's licenses issued by the state of Utah and the province of Quebec [19]. (Code 39 numbers have the same deficiency.) Since a Quebec driver's license number has twelve digits, notice that all errors in the third position are undetected! On the other hand, this scheme does detect 100% of transposition errors involving adjacent digits, while the UPC, IBM and most other modulo 10 schemes do not.

For check digits that satisfy a condition

$$(a_1, a_2, \ldots, a_n) \cdot (w_1, w_2, \ldots, w_n) \equiv 0 \pmod{k},$$

we may readily determine the undetectable single position errors and the undetectable transposition errors. (Actually, there is no compelling reason to use 0 in the condition. Any value in Z_k will do.)

Theorem 1 *Suppose an identification number $a_1 a_2 \ldots a_n$ satisfies*

$$(a_1, a_2, \ldots, a_n) \cdot (w_1, w_2, \ldots, w_n) \equiv 0 \pmod{k}.$$

Then a single-position error $a_i \to a_i'$ is undetectable if and only if $(a_i - a_i')w_i \equiv 0 \pmod{k}$ and a transposition error that interchanges the elements in the ith and jth positions is undetectable if and only if $(a_i - a_j)(w_i - w_j) \equiv 0 \pmod{k}$.

Proof. Consider a single error in the ith position. Say a_i' is substituted for a_i. Then the dot product for the correct number and the dot product for the incorrect number differ by $(a_i - a_i')w_i$. Thus the error is undetectable if and only if $(a_i - a_i')w_i \equiv 0 \pmod{k}$.

Now consider an error of the form

$$\ldots a_i a_{i+1} \ldots a_j a_{j+1} \ldots \to \ldots a_j a_{i+1} \ldots a_i a_{j+1} \ldots .$$

Then the dot product for the correct number and the dot product for the incorrect number differ by

$$(a_i w_i + a_j w_j) - (a_j w_i + a_i w_j) = (a_i - a_j)(w_i - w_j).$$

Thus, the error is undetectable if and only if

$$(a_i - a_j)(w_i - w_j) \equiv 0 \pmod{k}. \quad \blacksquare$$

When a check digit satisfies the condition $(a_1, a_2, \ldots, a_n) \cdot (w_1, w_2, \ldots, w_n) \equiv 0 \pmod{k}$ *and* the digits $a_1, a_2, \ldots, a_n$ are restricted to 0 to $k-1$ it is straightforward to determine conditions on the weights that ensure all errors of specific types are detectable. These are provided in Table 2.

Table 2. Conditions for detection of all errors of various types

Error type	Form	Condition for modulus k
single	$a_i \to a_i'$	$\gcd(w_i, k) = 1$
transposition	$\ldots a_i \ldots a_j \ldots \to \ldots a_j \ldots a_i \ldots$	$\gcd(w_j - w_i, k) = 1$
jump transposition	$abc \to cba$	$gcd(w_{i+2} - w_i, k) = 1$
twin	$aa \to bb$ (positions i and $i+1$)	$\gcd(w_i + w_{i+1}, k) = 1$
phonetic	$a0 \leftrightarrow 1a$ (positions i and $i+1$)	$jw_{i+1} \not\equiv (j-1)w_i$ for all $j = 0, 1, \ldots, k-1$
jump twin	$aca \to bcb$ (positions $i, i+1, i+2$)	$\gcd(w_i + w_{i+2}, k) = 1$

To apply the conditions given in Table 2 to methods such as the 3-weight scheme used by banks, notice that specifying that the check digit a_9 for the bank number $a_1a_2a_3a_4a_5a_6a_7a_8$ is $a_1a_2a_3a_4a_5a_6a_7a_8$ (mod 10) is equivalent to the condition $(a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9) \cdot (7, 3, 9, 7, 3, 9, 7, 3, 9) \equiv 0 \pmod{10}$.

The preceding theorem and Table 2 reveal the shortcomings of the various values for the modulus. In the case that the modulus k is less than 10, all single digit errors and all transposition errors cannot be detected without restricting the digits to range from 0 to $k-1$. For instance, serial numbers on airline tickets simply use the modulo 7 value of the number as the check digit. Since this scheme does not distinguish between a and a' when $|a-a'| = 7$ all such errors are undetectable. On the other hand, for modulus 11 schemes the conditions of the preceding theorem for the detection of all single errors and all transposition errors are simply met by choosing distinct weights between 0 and 10. As previously mentioned the only drawback of modulus 11 schemes is the necessity of introducing an extra character or the avoiding of certain numbers.

Noting that the schemes described above that utilize modulus 10 are not simultaneously 100% effective in detecting single digit errors and 100% effective in detecting transposition errors involving adjacent digits whereas the ones that use division by 11 are, it is natural to raise the question of whether it is possible to devise a one-check-digit scheme that utilizes division by 10 that detects all single-digit errors and all transposition errors involving adjacent digits. The next theorem says that the answer is no (see also [15]).

Theorem 2 [25] *Suppose an error detecting scheme with an even modulus detects all single-position errors. Then for every i and j there is a transposition error involving positions i and j that cannot be detected.*

Proof. Let the modulus be $2m$. Let us say that the digit x in position i contributes $\sigma_i(x)$ in the determination of the check digit. Obviously, in order to detect all single-position errors it is nec-

essary that the mappings σ_i be permutations. In order to detect all transposition errors involving positions i and j it is necessary that $\sigma_i(a) + \sigma_j(b) \neq \sigma_i(b) + \sigma_j(a)$ for all $a \neq b$ in Z_{2m}. It then follows that the mapping $\sigma(x) = \sigma_j(x) - \sigma_i(x)$ must be a permutation of Z_{2m}. But summing the elements of Z_{2m} modulo $2m$ we then have

$$m = m + 0 + (1 + 2m - 1) + (2 + 2m - 2) + \cdots + (m - 1 + m + 1).$$

Thus,

$$m = \sum x = \sum \sigma(x) = \sum(\sigma_j(x) - \sigma_i(x)) = \sum \sigma_j(x) - \sum \sigma_i(x) = m - m = 0.$$

This contradiction completes the proof. ∎

9. Non-commutative Schemes

In contrast to modulus 10 schemes that can not detect all single digit errors and all transposition errors of adjacent digits, in 1969 Verhoeff, in his Ph. D. thesis [42], devised a method utilizing a noncommutative algebraic system on the set $\{0, 1, \ldots, 9\}$ that detects all single-digit errors and all transposition errors involving adjacent digits without the necessity of introducing a new character as is the case for the ISBN method. (Interestingly, *after* Verhoeff devised his scheme two authors published "proofs" that a scheme that detected all such errors was not possible (see [42, p. 11]). Consider the permutation $\sigma(0) = 1, \sigma(1) = 5, \sigma(2) = 7, \sigma(3) = 6, \sigma(4) = 2, \sigma(5) = 8, \sigma(6) = 3, \sigma(7) = 0, \sigma(8) = 9, \sigma(9) = 4$ and the algebraic system defined by Table 3. The algebraic system defined in Table 3 is called the *dihedral group of order* 10 and is denoted by D_{10}.

The group D_{10} is known to chemists, geologists and physicists as the dihedral point group with 10 elements. Scientists use it to describe the 5 rotational symmetries and 5 reflectional symmetries of a regular pentagon. For example, the symbol 1 represents a rotation of 72 degrees whereas the symbol 5 represents a reflection across one of the five axes of symmetry. Then $1*5$ is a 72 degree rotation followed by the reflection represented by 5.

Table 5.2. Multiplication for D_5

$*$	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	0	6	7	8	9	5
2	2	3	4	0	1	7	8	9	5	6
3	3	4	0	1	2	8	9	5	6	7
4	4	0	1	2	3	9	5	6	7	8
5	5	9	8	7	6	0	4	3	2	1
6	6	5	9	8	7	1	0	4	3	2
7	7	6	5	9	8	2	1	0	4	3
8	8	7	6	5	9	3	2	1	0	4
9	9	8	7	6	5	4	3	2	1	0

Verhoeff's idea is to view the digits 0 to 9 as the elements of the group D_{10} and to replace ordinary addition with calculations done in D_{10}. In particular, to any string of digits $a_1 a_2 \ldots a_{n-1}$, we append the check digit a_n so that $\sigma(a_1) * \sigma^2(a_2) * \cdots * \sigma^{n-2}(a_{n-2}) * \sigma^{n-1}(a_{n-1}) * a_n = 0$. (Here $\sigma^i(x) = \sigma(\sigma^{i-1}(x))$. Since σ has the property that $\sigma^i(a) \neq \sigma^i(b)$ if $a \neq b$ all single digit errors are detected. Also, because

$$a * \sigma(b) \neq b * \sigma(a) \text{ if } a \neq b, \tag{1}$$

it follows that all transposition errors involving adjacent digits are detected (since (1) implies that $\sigma^i(a) * \sigma^{i+1}(b) \neq \sigma^i(b) * \sigma^{i+1}(a)$ if $a \neq b$). This scheme detects 95.6% of twin errors, 94.2% of jump transposition and jump twin errors and 100% of phonetic errors [42, p. 54]..

In 1990 the German Bundesbank began using the Verhoeff check digit scheme to append a check digit to the serial numbers on bank notes [34]. Table 4 gives the values of the functions $\sigma, \sigma^2, \ldots, \sigma^{10}$ needed for the computations. (The functional value $\sigma^i(j)$ appears in the row labeled with σ^i and the column labeled j.) Since the serial numbers on the bank notes are alphanumeric, it is necessary to assign numerical values to the letters. This assignment is shown in Table 5.

Table 5.3

	0	1	2	3	4	5	6	7	8	9
σ	1	5	7	6	2	8	3	0	9	4
σ^2	5	8	0	3	7	9	6	1	4	2
σ^3	8	9	1	6	0	4	3	5	2	7
σ^4	9	4	5	3	1	2	6	8	7	0
σ^5	4	2	8	6	5	7	3	9	0	1
σ^6	2	7	9	3	8	0	6	4	1	5
σ^7	7	0	4	6	9	1	3	2	5	8
σ^8	0	1	2	3	4	5	6	7	8	9
σ^9	1	5	7	6	2	8	3	0	9	4
σ^{10}	5	8	0	3	7	9	6	1	4	2

Table 5.4

A	D	G	K	L	N	S	U	Y	Z
0	1	2	3	4	5	6	7	8	9

To trace through a specific example consider the bank note (featuring the mathematician Gauss!) shown in Figure 4 with the number AG8536827U7. To verify that 7 is the appropriate check digit we observe that $\sigma(0) * \sigma^2(2) * \sigma^3(8) * \sigma^4(5) * \sigma^5(3) * \sigma^6(6) * \sigma^7(8) * \sigma^8(2) * \sigma^9(7) * \sigma^{10}(7) * 7 = 1 * 0 * 2 * 2 * 6 * 6 * 5 * 2 * 0 * 1 * 7 = 0$ as it should be. (To illustrate how to use the multiplication table for D_{10} we compute

$$1 * 0 * 2 * 2 = (1 * 0) * 2 * 2 = 1 * 2 * 2 = (1 * 2) * 2 = 3 * 2 = 0).$$

Figure 4. German bank note with serial number AG8536827U and check digit 7.

A shortcoming of the German bank note scheme is that it does not distinguish between a letter and its assigned numerical value. Thus, a substitution of 7 for U (or vice versa) and the transposition of 7 and U are not detected by the check digit. This shortcoming can be avoided by using D_{36}, the dihedral group of order 36, to assign every letter and digit a distinct value together with an appropriate permutation σ (see [20], [23] or [45]). Using this method to append one of the 36 alphanumeric characters as a check character, all single position errors and all transposition errors involving adjacent alphanumeric characters will be detected.

Although the error detecting schemes using non-commutative systems are more effective than the schemes that use modular arithmetic, the German bank note application is the only one I know that uses a non-commutative system.

We conclude with a comparison table of the error detection rates for the modulo 10 schemes and the scheme based on the dihedral group.

All of these schemes are 100% effective for detecting single digit errors. Recall that the detection rates for twin errors using 3-weight or 4-weight schemes depend on the length of the number. In the table we have used length 10. Winters [45] has devised a two check scheme based on the dihedral group of order 10 that detects 100% of all the errors listed in Table 6.

Table 6. Comparison of detection rates.

method	$ab \to ba$	$acb \to bca$	$aa \to bb$	$a0 \to 1a$	$aca \to bcb$
UPC	88.9	0	88.9	100	88.9
1,3,7, ...	88.9	88.9	55.6	100	66.7
7,3,1, ...	88.9	88.9	55.6	100	55.6
1,3,9,7, ...	88.9	88.9	88.9	100	0
1,3,7,9, ...	88.9	88.9	49.4	100	88.9
credit card	97.8	0	93.3	87.5	87.7
P.T.T.	96.3	96.3	95.6	95.8	95.6
dihedral	100	94.2	94.2	100	94.2

Acknowledgment. I am grateful to the referees for their comments. This article was written while the author was supported by the National Science Foundation (DMS-9225045) and the National Security Agency (MDA 904-91-H-0036).

References

1. Andrews, A.M. 1970. A variant of modulus 11 checking. *Comput. Bull.*, 14, 261-265.
2. Andrews, A.M. 1972. Decimal numbers with two check digits. *Comput. Bull.*, 16, 156-159.
3. Beckley, D. F. 1966. Check digit verification, *Data Processing*, 194-201.
4. Beckley, D.F. 1967. An optimum system with 'modulus 11'. *Comput. Bull.*, 11, 213-215.

5. Blocksma, M. 1989. *Reading the Numbers.* Penguin, New York.

6. Briggs, T. 1970. Modulus 11 check digit systems. *Comput. Bull.*, 14, 266-270.

7. Briggs, T. 1971. Weights for modulus 97 systems. *Comput. Bull.*, 15, 79.

8. Brown, D.A. H. 1973 Construction of error detection and correction codes to any base. *Electronic Letters*, 9, 290.

9. Brown, D.A. H. 1974. Biquinary decimal error detection codes with one, two and three check digits. *Comput. J.* , 17, 201-204.

10. Brown, D.A.H. 1974. Some error correcting codes for certain transposition and transcription errors in decimal integers. *Comput. J.*,17, 9-12.

11. Campbell, D.V.A. 1970. A modulus 11 check digit system for a given system of codes. *Comput. Bull.* 14, 12-13.

12. Chu, C.K. 1981. A note on multiple error detection in ASCII numeric data communication. *J. Ass. Comput. Mach.*, 28, 265-269.

13. Connor, S. 1984. The invisible border guard, *New Scientist*, 5, 9-14.

14. Dunn, Ashley. 1995. A pirate computer program builds credit card numbers. *New York Times*, Mar.19, 18.

15. Ecker, A. and Poch, G. 1986. Check character systems. *Computing*, 37, 277-301.

16. Fratini, S. 1989. Error detection in a class of decimal codes. *IEEE Trans. Inf. Theory*, 35, 1095-1098.

17. Freeman, H. 1967. Detection of transposition errors in decimal numbers. *Proc. IEEE*, 55, 1500.

18. Gallian, J.A. 1989. Check digit methods. *Inter. J. of Appl. Eng. Ed.*, 5, 503-505.

19. Gallian, J.A. 1991. Assigning driver's license numbers. *Math. Magazine*, 64, 13-22.

20. Gallian, J.A. 1991. The mathematics of identification numbers. *The Coll. Math. J.*, 22, 194-202.

21. Gallian, J. A. 1994. *Contemporary Abstract Algebra*, 3rd ed., D.C. Heath, Lexington.

22. Gallian, J.A. 1994. Coding information in *For all Practical Purposes*, 3rd ed., Freeman, New York.

23. Gallian, J. A. and Mullin, M. 1995. Groups with anti-symmetric mappings. *Archive der Math.*, to appear.

24. Gallian, J. A. and Winters, S. 1988. Modular arithmetic in the marketplace, *Amer. Math. Monthly*, 95, 548-551.

25. Gumm, H. P. 1985. A new class of check digit methods for arbitrary number systems, *IEEE Tran. Inf. Th.*, 31, 102-105.

26. Gumm, H. P. 1986. Encoding of numbers to detect typing errors, *Inter. J. Applied Eng. Ed.*, 2, 61-65.

27. Harmon, C. and Adams, R. 1989. *Reading Between the Lines.* Helmers Publishing, NH.

28. Hill, R. 1986. *A First Course in Coding Theory.* Clarendon Press, Oxford.

29. Kunerth, W. 1981. Die EDV-gerechte Verschlüsselung: Grundlagen und Anwendungen moderner Nummernsysteme. Stuttgart: Forkel.

30. Larsen, H.L. 1983. Generalized double modulus 11 check digit error detection. *BIT*, 23, 303-307.

31. Pezzulo, J. 1989. Letter to the editor. *Comm. ACM*, 32, 1131-1132.

32. Schechinger, H. 1979. Kontrolle mit Hilfe von Prüfziffern, *Burotechnische Sammlung, BTS systemtisch*, 171.

33. Schulz, R.-H. 1991. Some check digit systems over non-abelian groups. *Mitt. Math. Ges. Hamburg*, 12, 819-827.

34. Schulz, R.-H. 1991. *Codierungstheorie. Eine Einführung.* Braunschweig-Wiesbaden.

35. Selmer, E.S. 1967. Registration numbers in Norway: some applied number theory and psychology. *J. Royal Statistical Soc., Ser A*, 130, 225-231.

36. Sethi, A.S., Rajaraman, V. and Kenjale, P.S. 1978. An error correcting scheme for alphanumeric data. *Inform. Process. Lett.*, 7, 72-77.

37. Sisson, R. L. 1958. An improved decimal redundancy check, *Comm. ACM*, 1, 10-12.

38. *Standard Book Numbering.* 1968, The Standard Book Numbering Agency LTD, London.

39. Tang, D.T. and Lum, V.Y. 1970. Error control for terminals with human operators. *IBM J. Res. Develop.*, 14, 409-416.

40. *UPC Symbol Specification Manual.* 1986. Uniform Code Council, Dayton, Ohio.

41. *USS-128 Uniform Symbology Specification.* 1986. Automatic Identification Manufacturers, Pittsburgh, PA.

42. Verhoeff, J. 1969. *Error Detecting Decimal Codes.* Mathematical Centre, Amsterdam.

43. Wagner, N. R. and Putter, P. S. 1989. Error detecting decimal digits, *Comm. ACM*, 32, 106-110.

44. Wild, W.G. 1968. The theory of modulus N check digit systems. *Comput. Bull.*, 12, 309-311.

45. Winters, S. 1990. Error detecting codes using dihedral groups. *UMAP Journal*, 11, 299-308.