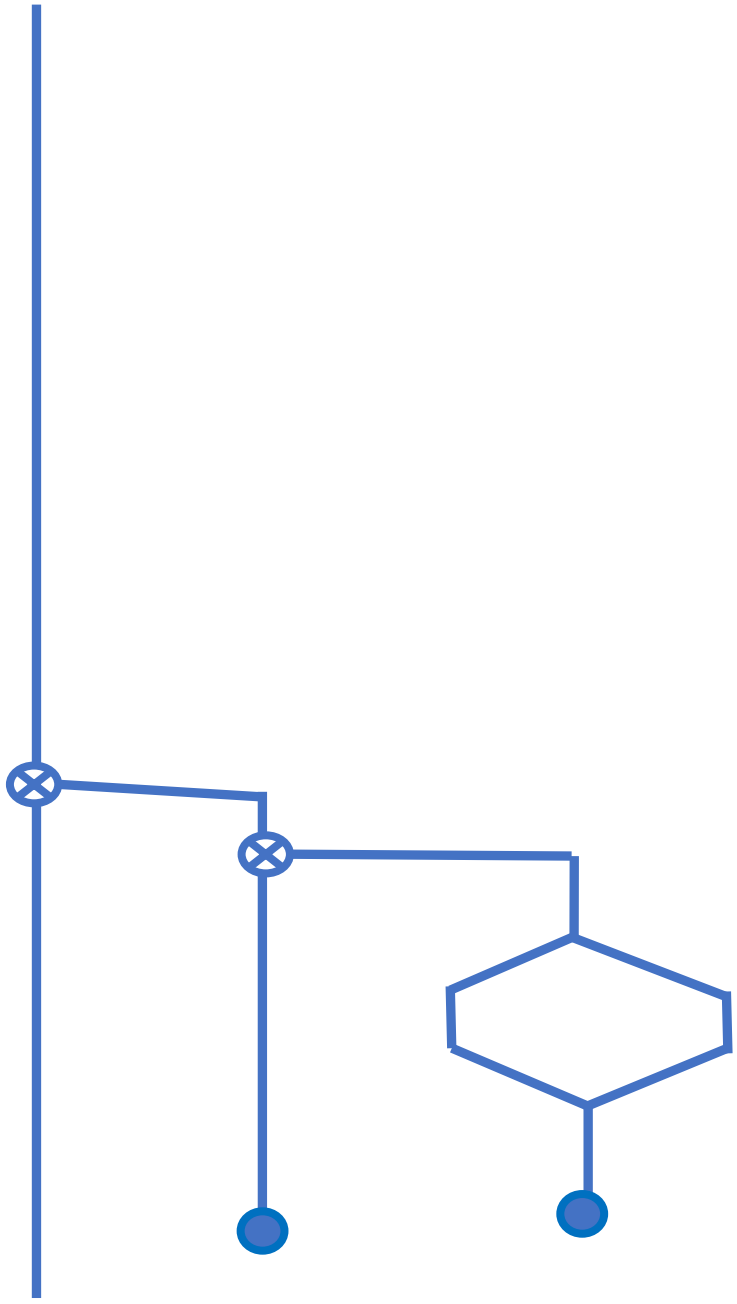


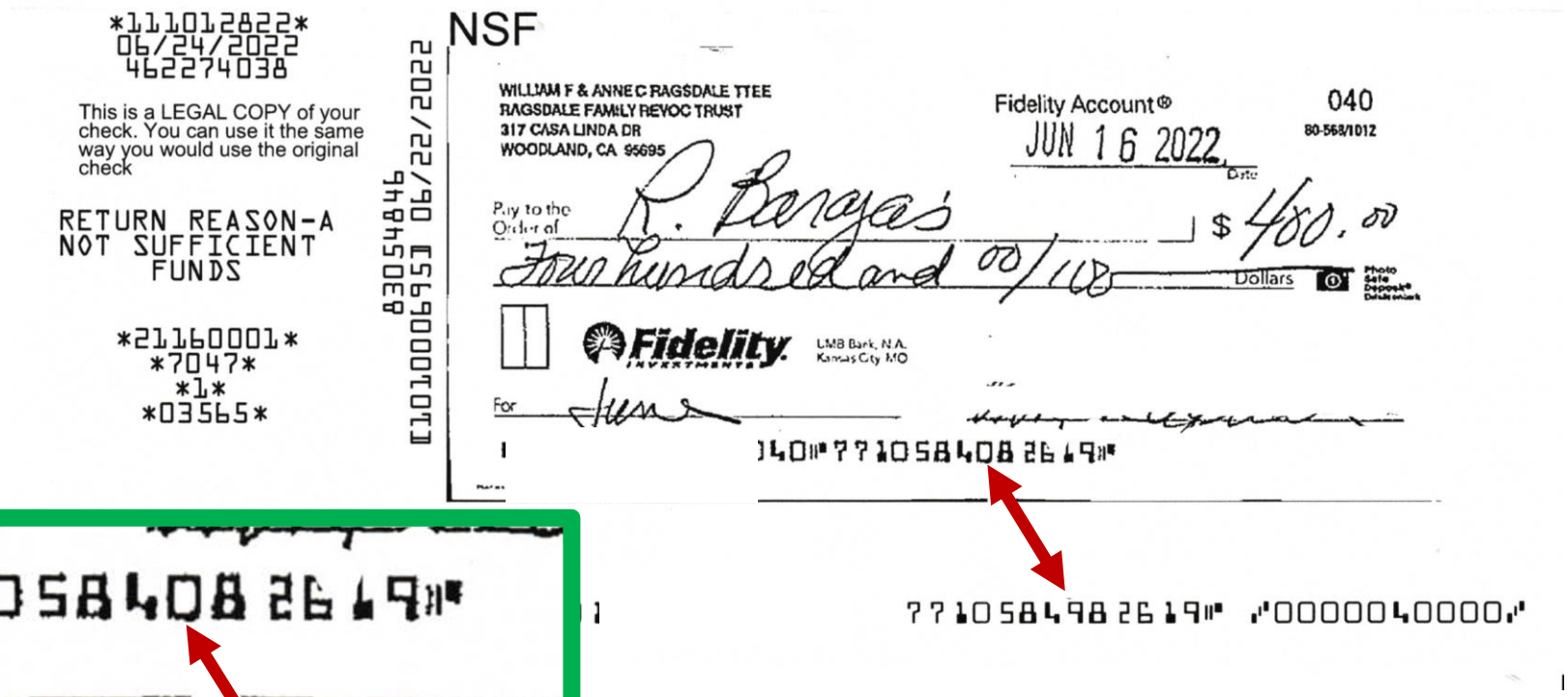
Checksum Challenge

SVFIG
Aug. 27, 2022
Bill Ragsdale

The Challenge

1. Create a one or two-digit check code for a 10-digit account number.
2. Detect a single digit in error.
3. Detect the exchange of two adjacent digits.

How It All Started



Substitution Algorithm

1. Decompose a 10 digit account number into ten decimal bytes.
2. Compute sum of the ten bytes, modulo 10.
3. Append that check sum as an eleventh digit of the account number.
4. Upon repeated readings of the account number, check the sum of account digits matches the check digit.

Setup

```
0 VALUE Reference.Number      \ our input value
0 VALUE Test.Number           \ this one varies
CREATE 10.digits 10 allot     \ byte expansion
\   decade check digit:
0 VALUE reference.decade.sum \ modulo result
0 VALUE      test.decade.sum
\   position check digits
0 VALUE reference.position.check
0 VALUE      test.position.check
```

Expand Into 10 Bytes

```
: Build.10.digits ( acct --- )
   \ creates 10.digits array
    -1 9 do                    \ declining loop
          10 /mod swap    \  quotient remainder
          10.digits i + c!
          -1 +loop drop ;
```

Checksum Calculation

```
\ Create decade.sum

: From.10.digits  ( --- mod.result )
    \ compute mod 10 of 10.digits

       0

  10 0 do  10.digits i + c@  +  loop

      10 mod    ;
```

Driving Program

```
: Validate.decade.check ( -- )
    Reference.Number Build.10.digits
    From.10.digits to reference.decade.sum

    Test.Number Build.10.digits
    From.10.digits to test.decade.sum

\ Test for matching check sums
   reference.decade.sum test.decade.sum = cr cry
   if   ." Decade check sums match. “
    else ." Error in decade check sums." then  ;
```

Substitution Error

1234567890

1 + 1
2 + 3
3 + 6
4 + 10
5 + 15
6 + 21
7 + 28
8 + 36
9 + 45
0 + 45
10 mod = 5

1234567891

1 + 1
2 + 3
3 + 6
4 + 10
5 + 15
6 + 21
7 + 28
8 + 36
9 + 45
1 + 46
10 mod = 6

Digit Error Example

```
1234567890 to Reference.number
1234567891 to Test.number
Validate.decade.check


1234567890
1234567891
Reference check sum is  5
Test check sum is  6
Error in decade check sums.
```

Transposition Algorithm

1. Decompose a 10 digit account number into ten decimal bytes.
2. Over the ten digits, at an even position add the digit. At an odd position, subtract the digit.
3. Append that check sum as an eleventh digit of the account number.
4. Upon repeated readings of the account number, check the sum of account digits matches the check digit.

Transposition Error

1234567890		
1	+	1
2	-	-1
3	+	2
4	-	-2
5	+	3
6	-	-3
7	+	4
8	-	-4
9	+	5
0	-	5

1234657890		
1	+	1
2	-	-1
3	+	2
4	-	-2
6	+	4
5	-	-1
7	+	6
8	-	-2
9	+	7
0	-	7

Check Digit Calculation

```
: Position.check ( --- value )
\ over 10.digits build position test value
    0
    10 0 do
        10.digits i     + c@   +  \ even position
        10.digits i 1+  + c@   -  \  odd position
        2 +loop
     10 mod  ;
```

Driving Program

```
: Validate.positions
  Reference.number build.10.digits
  Position.check to reference.position.check
  Test.number build.10.digits
  Position.check to test.position.check
\ Check matching position digit
   reference.position.check
         cr ." Reference check is " dup .
   test.position.check
            ." Test check is "      dup .
   = if  cr ." Position checks match. “
      else cr ." Error in postion checks." then
   ;
```

Transposition Example

1234567890

1234657890

Reference check is 5 Test check is 7

Error in position checks. ok

Summary

Credit card magnetic stripes use the decimal check sum method plus a parity bit on each digit.

By xor-ing the digits of the two check methods, it would be possible to detect either error alone.

Summary

However, if multiple errors occur, these methods will only trap 9 of every 10 errors. Not very good odds.

These methods assume a high degree of reading accuracy.

AFAIK There are no error checks on the MICR coding on traditional paper checks.

Incidentally, check processing is a very dirty, lint filled environment.