

One and the Same Check Digit Scores Anywhere From 0 to 1 Depending on the Kind of Error

— Over **400** eight-digit numbers with every single-digit slip applied, all three schemes catch them at a share of **1.000000** — the separator is whether one asks about a single-digit slip or about a swap — [Paper 845]

Yuuki YAMAGISHI (山岸 勇輝)

Yamagishi Complexity Torus Lab, Japan

September 2026

YOI: <https://www.yamagishiyuuki.com/yoi/845>

— *From thinking to syncing.* —

Abstract

A final digit is said to “protect” a number. This paper counts that the score runs from 0 to 1 depending on what it protects against. No new theorem or law is claimed. First, single-digit slips are caught by every scheme: all three catch them at a share of **1.000000** (Section 2). Second, and this is the core. On swaps the schemes split: the plain sum catches **0.000000** and the weighted scheme **1.000000**: both ends, for one and the same single check digit (Section 3). Third, the plain sum’s 0 is no accident. It never uses the order of the terms, so rearranging leaves the sum untouched (Section 3). Fourth, Luhn sits between them: catching a share of **0.979503** and letting the rest through (Section 3). Fifth, only one kind gets through: all **52** escapes were swaps of **0** against **9** (Section 3). Sixth, the separator is whether one asks about a single-digit slip or about a swap: “protects” fixes no meaning until it says against what (Section 4). **Keywords:** check digit, error detection, adjacent swap, weights, modulus, exhaustive enumeration.

Scope of this paper (scope note): No new theorem or law is claimed: that a check digit catches all single-digit errors, that catching swaps requires weights that vary by position, that the Luhn scheme misses only 0 against 9, and that a weighted scheme modulo 11 catches both kinds, are all known. The use of real numbering systems is not discussed, which standard to adopt, and how to migrate, are not stated. Error correction is not entered: repairing the wrong digit is not treated; only detection. The frequency of slips is not treated: how often people make each kind of mistake is not included; all kinds are counted equally. Simultaneous errors are not counted: cases where two or more digits change at once were not counted. Relation to earlier papers: Paper 752 showed that the Huffman code is optimal, but only within the constraint that code lengths be whole numbers. This paper stands beside it: there which constraint hides behind “optimal”, here which error hides behind “protects”. Paper 300 showed that whether two things share a root is decidable: single-digit slips and swaps are distinct roots by that criterion. What is added is laying three schemes over one sample and measuring both kinds of error, counting that the scores split from 0.000000 to 1.000000, pinning Luhn’s escapes to the single kind 0 against 9, and placing the separator on slip versus swap.

1. Notation and the Question

Setting: for a string of eight digits, three schemes compute a check digit.

$$\text{plain sum} = \sum_i d_i \bmod 10, \quad \text{weighted} = \sum_i w_i d_i \bmod 11$$

The Luhn scheme doubles every second digit from the right, folding two-digit results by adding their digits, then sums. An error counts as caught when the check digit changes.

Question: “protects” against what?

2. Single-Digit Slips Are Caught by All

Computation 2.1: 400 eight-digit numbers were built and every digit replaced by each of the other nine (28800 cases).

scheme	share caught
plain sum (modulo 10)	1.000000
Luhn (modulo 10)	1.000000
weighted (modulo 11)	1.000000

Observation 2.2

all three caught everything. Changing one digit always changes that digit's contribution: with no weight equal to zero, the sum must move.

Observation 2.3

so looking here alone the three appear identical. "Detects single-digit errors" does not separate them: the difference only appears on another kind of error.

3. On Swaps They Split

Computation 3.1: on the same 400 numbers, every swap of adjacent digits was applied.

scheme	share caught
plain sum (modulo 10)	0.000000
Luhn (modulo 10)	0.979503
weighted (modulo 11)	1.000000

Observation 3.2

this is the core. One single check digit produces both ends, 0 and 1. The plain sum catches nothing and the weighted scheme lets nothing through, because the plain sum never uses the order of the terms, so rearranging leaves it unchanged.

Observation 3.3

this 0.000000 is not an empty swing. On the same sample, single-digit slips were caught at 1.000000: same scheme, same sample; only the kind of error changed, and 1 became 0.

Computation 3.4: Luhn's escapes were counted by which pair of digits was swapped.

pair that escaped	cases
0 against 9	52
anything else	0

Observation 3.5

only one kind gets through. Only for 0 against 9 does doubling-and-folding leave the sum unchanged: the awkward 0.979503 is just the share this one kind occupies.

Observation 3.6

the hole can be named because the score can be measured. The share alone says “nearly everything”; splitting the escapes narrows it to one kind, where it is weak and how weak it is are different pieces of information.

4. The Separator

Observation 4.1

the separator is whether one asks about a single-digit slip or about a swap. On the first the three schemes stand level; on the second they split from 0 to 1: “protects” fixes no meaning until it says against what.

Observation 4.2

placed beside Paper 752.752 showed that the Huffman code is optimal, but only within the constraint that code lengths be whole numbers. There which constraint hides behind “optimal”; here which error hides behind “protects”. Both bring out a condition buried under an adjective. Paper 300 showed that whether two things share a root is decidable: the two kinds of error are distinct roots by that criterion.

Observation 4.3

what is not said. Choosing a real standard, error correction and the frequency of slips are not treated. What is shown is that one digit scores 0 or 1 according to the kind of error.

5. Census and Closing

1. slips: all three at **1.000000** (Computation 2.1): **indistinguishable**.
2. **28800** cases, all of them (Computation 2.1): **not a sample**.
3. plain sum on swaps: **0.000000** (Computation 3.1): **the core**.
4. weighted: **1.000000** (Computation 3.1): **the other end**.
5. Luhn: **0.979503** (Computation 3.1): **in between**.
6. same sample, 1 became 0 (Observation 3.3): **no empty swing**.
7. **52** escapes (Computation 3.4): **only 0 against 9**.
8. everything else: **0** (Computation 3.4): **the hole is named**.
9. where weak against how weak (Observation 3.6): **different information**.
10. slip, or swap (Observation 4.1): **the separator; Papers 752, 300**.

Closing: One and the same check digit scores anywhere from 0 to 1 depending on the kind of error. Applying all 28800 single-digit slips to 400 eight-digit numbers, all three schemes catch them at 1.000000 and cannot be told apart, because no weight is zero, so changing a digit must move the sum. Yet on swaps of adjacent digits the plain sum catches 0.000000, Luhn 0.979503 and the weighted scheme 1.000000, because the plain sum never uses the order of the terms, so rearranging leaves it unchanged. All 52 of Luhn's escapes were swaps of 0 against 9, so the hole narrows to one kind. Placed among the earlier papers: beside Paper 752's "optimal, but within which constraint", as "protects, but against which error", and treating the two kinds of error as distinct roots by the criterion of Paper 300. To be honest: all kinds were counted equally, with nothing included about how often people actually make each mistake.

References

1. Kirtland, J. (2001). *Identification Numbers and Check Digit Schemes*. Mathematical Association of America.
2. Yamagishi, Y. (2026). *The Three Hundredth Census: Whether Two Things Share a Root Is Decidable* [Paper 300]. YOI: <https://www.yamagishiyuuki.com/yoi/300>; DOI: 10.5281/zenodo.22171914

3. Yamagishi, Y. (2026). *The Huffman Code Is Optimal, but Only Within the Constraint That Code Lengths Be Whole Numbers* [Paper 752]. YOI: <https://www.yamagishiyuuki.com/yoi/752>; DOI: 10.5281/zenodo.23015146

— *From thinking to syncing.* —

Yuuki YAMAGISHI (山岸 勇輝)

© 2026 Yamagishi Complexity Torus Lab, Japan · Licensed under Creative Commons Attribution 4.0 International (CC BY 4.0)