WIKIPEDIA

Luhn algorithm

The **Luhn algorithm** or **Luhn formula**, also known as the "modulus 10" or "mod 10" algorithm, named after its creator, IBM scientist Hans Peter Luhn, is a simple check digit formula used to validate a variety of identification numbers.

It is described in U.S. Patent No. 2,950,048, granted on August 23, 1960.[1]

The algorithm is in the public domain and is in wide use today. It is specified in ISO/IEC 7812-1.[2] It is not intended to be a cryptographically secure hash function; it was designed to protect against accidental errors, not malicious attacks. Most credit cards and many government identification numbers use the algorithm as a simple method of distinguishing valid numbers from mistyped or otherwise incorrect numbers.

Description

The check digit is computed as follows:

1. If the number already contains the check digit, drop that digit to form the "payload". The check digit is most often the last digit.
2. With the payload, start from the rightmost digit. Moving left, double the value of every second digit (including the rightmost digit).
3. Sum the values of the resulting digits.
4. The check digit is calculated by $(10 - (s \bmod 10)) \bmod 10$, where s is the sum from step 3. This is the smallest number (possibly zero) that must be added to s to make a multiple of 10. Other valid formulas giving the same value are $9 - ((s + 9) \bmod 10)$, $(10 - s) \bmod 10$, and $10\lceil s/10 \rceil - s$. Note that the formula $(10 - s) \bmod 10$ will not work in all environments due to differences in how negative numbers are handled by the modulo operation.

Example for computing check digit

Assume an example of an account number 1789372997 (just the "payload", check digit not yet included):

	7	9	9	2	7	3	9	8	7	1
Multipliers	2	1	2	1	2	1	2	1	2	1
	=	=	=	=	=	=	=	=	=	=
	14	9	**18**	2	**14**	3	**18**	8	**14**	1
Sum digits	**5** (1+4)	9	**9** (1+8)	2	**5** (1+4)	3	**9** (1+8)	8	**5** (1+4)	1

The sum of the resulting digits is 56.

The check digit is equal to $10 - (56 \bmod 10) = 4$.

This makes the full account number read 17893729974.

Example for validating check digit

1. Drop the check digit (last digit) of the number to validate. (e.g. 17893729974 → 1789372997)

2. Calculate the check digit (see above)
3. Compare your result with the original check digit. If both numbers match, the result is valid. (e.g. (givenCheckDigit = calculatedCheckDigit) ⇔ (isValidCheckDigit)).

Strengths and weaknesses

The Luhn algorithm will detect all single-digit errors, as well as almost all transpositions of adjacent digits. It will not, however, detect transposition of the two-digit sequence *09* to *90* (or vice versa). It will detect most of the possible twin errors (it will not detect *22* ↔ *55*, *33* ↔ *66* or *44* ↔ *77*).

Other, more complex check-digit algorithms (such as the Verhoeff algorithm and the Damm algorithm) can detect more transcription errors. The Luhn mod N algorithm is an extension that supports non-numerical strings.

Because the algorithm operates on the digits in a right-to-left manner and zero digits affect the result only if they cause shift in position, zero-padding the beginning of a string of numbers does not affect the calculation. Therefore, systems that pad to a specific number of digits (by converting 1234 to 0001234 for instance) can perform Luhn validation before or after the padding and achieve the same result.

The algorithm appeared in a United States Patent[1] for a simple, hand-held, mechanical device for computing the checksum. The device took the mod 10 sum by mechanical means. The *substitution digits*, that is, the results of the double and reduce procedure, were not produced mechanically. Rather, the digits were marked in their permuted order on the body of the machine.

Pseudocode implementation

The following function takes a card number, including the check digit, as an array of integers and outputs **true** if the check digit is correct, **false** otherwise.

```
function isValid(cardNumber[1..length])
    sum := 0
    parity := length mod 2
    for i from 1 to length do
        if i mod 2 != parity then
            sum := sum + cardNumber[i]
        elseif cardNumber[i] > 4 then
            sum := sum + 2 * cardNumber[i] - 9
        else
            sum := sum + 2 * cardNumber[i]
        end if
    end for
    return cardNumber[length] == (10 - (sum mod 10))
end function
```

Code implementation

C#

```
bool IsValidLuhn(in int[1] digits)
{
    int check_digit = 0;
    for (int i = digits.Length - 2; i >= 0; --i)
        check_digit += ((i & 1) is 0) switch
        {
            true  => digits[i] > 4 ? digits[i] * 2 - 9 : digits[i] * 2,
            false => digits[i]
        };
```

```
    return 10 - (check_digit % 10) == digits.Last();
}
```

Java

```java
public static boolean isValidLuhn(String number) {
        int checksum = Character.getNumericValue(number.charAt(number.length() - 1));
        int total = 0;

        for (int i = number.length() - 2; i >= 0; i--) {
            int sum = 0;
            int digit = Character.getNumericValue(number.charAt(i));
            if (i % 2 == 0) {
                digit *= 2;
            }

            sum = digit / 10 + digit % 10;
            total += sum;
        }

        return 10 - total % 10 == checksum;
    }
```

Uses

The Luhn algorithm is used in a variety of systems:

- Credit card numbers
- IMEI numbers
- National Provider Identifier numbers in the United States
- Canadian social insurance numbers
- Israeli ID numbers
- South African ID numbers
- South African Tax reference numbers
- Swedish national identification numbers
- Swedish Corporate Identity Numbers (OrgNr)
- Greek Social Security Numbers (AMKA)
- ICCID of SIM cards
- European patent application numbers
- Survey codes appearing on McDonald's, Taco Bell, and Tractor Supply Co. receipts
- United States Postal Service package tracking numbers use a modified Luhn algorithm[3]

References

1. US patent 2950048A (https://worldwide.espacenet.com/textdoc?DB=EPODOC&IDX=US2950048A), Luhn, Hans P., "Computer for verifying numbers", published 1960-08-23
2. "Annex B: Luhn formula for computing modulus-10 "double-add-double" check digits". *Identification cards — Identification of issuers — Part 1: Numbering system* (https://www.iso.org/standard/70484.html) (Standard). International Organization for Standardization, International Electrotechnical Commission. January 2017. ISO/IEC 7812-1:2017.
3. "Publication 199: Intelligent Mail Package Barcode (IMpb) Implementation Guide for Confirmation Services and Electronic Payment Systems" (https://postalpro.usps.com/pub199). United States Postal Service. Retrieved 29 November 2023.

External links

- Implementation in 150 languages on the Rosetta Code project (https://rosettacode.org/wiki/Luhn_test_of_credit_card_numbers)