Aplikasi Aritmetika Modulo dalam Validasi Nomor Kartu Kredit

Yudha Adiprabowo - 13506050
Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if16050@students.if.itb.ac.id

Abstract – *Makalah ini membahas tentang aritmetika modulo dan pemakaiannya dalam validasi nomor kartu kredit. Untuk membuat sebuah nomor kartu kredit yang sahih, dibutuhkan sebuah algoritma yang standard dan bisa dipakai dengan mudah, yaitu algoritma Luhn, yang menggunakan aritmetika modulo dalam operasinya. Makalah ini bukan bertujuan untuk melakukan* brute-force *check untuk mencari nomor-nomor kartu kredit yang sahih untuk kemudian digunakan untuk bertransaksi secara ilegal di internet, tetapi lebih untuk tujuan pengetahuan umum.*

Kata Kunci*: aritmetika modulo, algoritma Luhn, kartu kredit*

1. PENDAHULUAN

Kartu kredit pada masa sekarang banyak digunakan untuk melakukan transaksi perbankan secara online. Salah satu syarat untuk melaksanakan transaksi tersebut adalah, kita harus memasukkan nomor kartu kredit kita dalam situs tempat kita melakukan transaksi. Sebenarnya, angka-angka yang menyusun nomor kartu kredit bukanlah angka biasa. Angka-angka tersebut merupakan angka yang telah melewati pemeriksaan menggunakan algoritma Luhn, yang memanfaatkan aritmetika modulo.

2. ARITMETIKA MODULO

2.1. Definisi Aritmetika Modulo

Aritmetika Modulo (kadang disebut juga aritmetika jam) adalah sistem aritmetika untuk bilangan bulat di mana kedua bilangan bulat dioperasikan sampai mencapai nilai tertentu, yaitu modulus (sisa). Aritmetika Modulo diperkenalkan oleh Carl Friedrich Gauss dalam bukunya *Disquistiones Arithmaticae* yang dipublikasikan pada tahun 1801.

Contoh sederhana penggunaan aritmetika modulo terdapat pada sistem 24-jam. Dalam satu hari, mulai dari tengah malam sampai tengah malam, terdapat 24 jam yang dimulai dari 0 sampai 23. Jika sekarang pukul 20.00, maka 9 jam lagi adalah pukul 05.00, bukan pukul 29.00. Karena angka jam direset kalau mencapai 24, aritmetika jam adalah aritmetika *modulo 24*. Perlu diingat bahwa 24.00 bukanlah angka jam yang valid karena sama dengan pukul 00.00, sama halnya dengan 02.60 yang sama dengan 03.00.

Misalkan a adalah bilangan bulat dan m adalah bilangan bulat > 0. Operasi a mod m (dibaca "a modulo m") memberikan sisa jika a dibagi dengan m. Dengan kata lain, a mod $m = r$ sedemikian sehingga $a = mq + r$ dengan $0 \leq r < m$.

Bilangan m adalah yang disebut modulus atau modulo, dan hasil aritmetika modulo m terletak di dalam himpunan {0, 1, 2, ..., m-1}. Beberapa contoh operasi dengan operator modulo yaitu: 6 mod 8 = 6 (karena 6 = 8 · 0 + 6), -39 mod 13 = 0 (karena -39 = 13(-2) + 0).

Jika a mod m = 0, maka dikatakan bahwa a adalah kelipatan dari m, yaitu a habis dibagi dengan m. Misalnya, 54 mod 6 = 0, berarti 54 adalah kelipatan 6.

2.2 Kongruen

Kadang-kadang dua buah bilangan bulat, a dan b, mempunyai sisa yang sama jika dibagi dengan bilangan bulat positif m. Kita katakan bahwa a dan b kongruen dalam modulo m, dan dilambangkan sebagai

$$a \equiv b \ (mod\ m) \qquad (2.1)$$

(notasi '≡' dibaca 'kongruen')

Jika a tidak kongruen dengan b dalam modulus m, maka ditulis

$$a \not\equiv b \ (\text{mod } m) \qquad (2.2)$$

Misalnya 38 mod 5 = 3 dan 13 mod 5 = 3, maka 38 ≡ 13 (mod 5). Hal yang sama juga berlaku untuk negatif dari a, -38 ≡ 13 (mod 5). Definisi formal dari kekongruenan dinyatakan sebagai berikut:

Misalkan a dan b adalah bilangan bulat dan m adalah bilangan > 0, maka $a \equiv b$ (mod m) jika m habis membagi a-b.

Kekongruenan $a \equiv b$ (mod m) dapat pula dituliskan dalam hubungan

$$a = b + km \qquad (2.3)$$

yang dalam hal ini k adalah bilangan bulat. Kita juga dapat menuliskan a mod $m = r$ sebagai

$$a \equiv r \ (mod\ m) \qquad (2.4)$$

Beberapa sifat dari pengerjaan hitung pada aritmetika

modulo adalah sebagai berikut:

Jika $a_1 \equiv b_1 \pmod{n}$ dan $a_2 \equiv b_2 \pmod{n}$ maka:

- $(a_1 + a_2) \equiv (b_1 + b_2) \pmod{n}$
- $(a_1 - a_2) \equiv (b_1 - b_2) \pmod{n}$
- $(a_1 a_2) \equiv (b_1 b_2) \pmod{n}$.

Jika $a \equiv b \pmod{m}$ dan c adalah sembarang bilangan bulat maka:

- $(a + c) \equiv (b + c) \pmod{m}$
- $ac \equiv bc \pmod{m}$
- $a^p \equiv b^p \pmod{m}$ untuk suatu bilangan bulat tak negatif p.

2.3. Algoritma Luhn

Algoritma Luhn atau Formula Luhn, dikenal juga sebagai algoritma "modulus 10", adalah sebuah formula *checksum* sederhana yang digunakan untuk memvalidasi macam-macam Nomor Identifikasi seperti nomor Kartu Kredit. *Checksum* adalah sebuah cara sederhana untuk melindungi integritas data dengan mendeteksi kesalahan (*error*) dalam data yang dikirim melalui media telekomunikasi atau data dalam media penyimpanan.

Algoritma ini diciptakan oleh peneliti IBM Hans Peter Luhn. Saat ini algoritma Luhn bebas digunakan publik dan telah dipakai secara luas. Algoritma Luhn tidak didesain untuk menjadi sebuah fungsi hash yang aman secara kriptografi. Algoritma ini lebih didesain untuk melindungi dari sebuah kesalahan tak sengaja pada data, bukan dari serangan dengan maksud mencuri data. Banyakan kartu kredit dan nomor identifikasi pemerintah menggunakan algoritma ini sebagai cara sederhana untuk membedakan nomor yang sahih dari nomor-nomor acak.

Cara kerja algoritma ini dapat dijelaskan sebagai berikut:

- Mulai dari angka paling kanan sampai paling kiri, kalikan setiap angka kedua dengan 2. Untuk semua angka yang menjadi lebih besar dari atau sama dengan 10, kurangi angka hasil perkalian tadi dengan 9. Sebagai contoh, 1111 menjadi 2121, sementara 8763 menjadi 7733 (dari 2x6 = 12 lalu 12-9 = 3 dan 2x8 = 16 lalu 16-9 = 7).
- Jumlahkan semua angka yang ada. Sebagai contoh, jika 1111 menjadi 2121, maka 2+1+2+1 = 6; dan 8763 menjadi 7733, maka 7+7+3+3 = 20.
- Jika total dari penjumlahan tadi dimoduluskan 10 kongruen dengan 0, maka nomor itu sahih. Jadi, 1111 bukan angka yang sahih (6 mod 10 ≠ 0) sementara 8763 adalah sahih (20 mod 10 = 0).

Tabel 1: Pemetaan angka dalam algoritma Luhn

Angka	x2	-9	Hasil
0	0	0	0
1	2	2	2
2	4	4	4
3	6	6	6
4	8	8	8
5	10	10-9	1
6	12	12-9	3
7	14	14-9	5
8	16	16-9	7
9	18	18-9	9

Tabel 1 menunjukkan pemetaan setiap angka dalam algoritma Luhn. Dapat dilihat bahwa 0123456789 dipetakan menjadi 0246813579.

3. VALIDASI NOMOR KARTU KREDIT

3.1 *Major Industry Indentifier*

Angka pertama dari nomor kartu kredit adalah tanda pengenal industri mayor / *Major Industry Identifier* (MII), yang merepresentasikan kategori perusahaan yang mengeluarkan kartu kredit tersebut. MII yang berbeda-beda merepresentasikan kategori seperti dalam tabel 2.

Tabel 2: Kategori dalam MII

Angka MII	Kategori perusahaan
0	ISO/TC 68 dan industri lain-lain
1	Perusahaan penerbangan
2	Perusahaa penerbangan dan industri lain-lain
3	Travel dan hiburan
4	Bank dan finansial
5	Bank dan finansial
6	*Merchandizing* dan bank
7	Perusahaan petroleum
8	Telekomunikasi dan industri lain-lain
9	Perusahaan negara

Sebagai contoh, American Express, Diner's Club, dan Carte Blanche berada di kategori travel dan hiburan, VISA, MasterCard, dan Discover berada di kategori bank dan finansial, sementara SUN Oil dan Exxon berada di kategori perusahaan petroleum.

3.2. Identifikasi perusahaan yang mengeluarkan kartu kredit

Enam angka awal dari nomor kartu kredit (termasuk angka MII) membentuk *issuer identifier* (pengenal perusahaan yang mengeluarkan kartu kredit). Ini berarti total probabilitas perusahaan yang mengeluarkan kartu kredit adalah satu juta kemungkinan (10 dipangkatkan 6).

Beberapa perusahaan terkenal dijelaskan pada tabel 3.

Tabel 3: Beberapa perusahaan terkenal yang mengeluarkan kartu kredit

Nama Perusahaan	*Identifier*	Panjang angka kartu
Diner's Club/Carte	300xxx-	14

Blanche	305xxx, 36xxxx, 38xxxx	
American Express	34xxxx, 37xxxx	15
VISA	4xxxxx	13, 16
MasterCard	51xxxx- 55xxxx	16
Discover	6011xx	16

Jika angka MII adalah 9, maka 3 angka berikutnya adalah kode negara yang didefinisikan dalam ISO 3166, dan sisa 2 angkanya bisa didefinisikan oleh badan standar negara masing-masing sesuai kebijakan dari negara tersebut.

3.3. Nomor Rekening

Angka 7 hingga (n – 1) dari nomor kartu kredit adalah tanda pengenal rekening Anda. Panjang maksimum dari sebuah nomor kartu kredit adalah 19 digit. Berhubung 6 angka pertama adalah *issuer identifier* dan angka terakhir adalah angka untuk mengecek, panjang maksimum dari sebuah nomor rekening adalah 19 – 7 digit yaitu 12 digit. Setiap perusahaan yang mengeluarkan kartu kredit berarti memiliki 1 trilyun kemungkinan nomor rekening (10 pangkat 12 atau 1.000.000.000.000).

3.4. Contoh Validasi Nomor Kartu Kredit

Sebagai contoh, kita ambil sebuah kartu VISA dengan nomor 4408 0412 3456 7890.

Major Industry Identifier (MII) adalah 4 (kategori bank dnd finansial), *issuer identifier* adalah 440804 (partner dari VISA), nomor rekeningnya adalah 123456789, dan angka untuk mengecek adalah 0.

Mari kita coba pengecekan dengan algoritma Luhn pada 4408 0412 3456 7890. Pada tabel 4 berikut,

- Baris teratas adalah angka sebenarnya.
- Baris kedua adalah x2.
- Baris ketiga adalah -9 jika dibutuhkan.
- Baris keempat adalah hasil akhir

Jika kita jumlahkan semua angka di baris keempat, kita mendapatkan hasil 67. 67 mod 10 ≠ 0. Sehingga bisa kita simpulkan bahwa 4408 0412 3456 7890 bukan merupakan angka sahih untuk nomor kartu kredit.

Dengan mengganti angka terakhir (angka untuk mengecek) dari 0 menjadi 3, kita mendapatkan angka 4408 0412 3456 7893, yang melewati pemeriksaan ini, karena hasil penjumlahan dari angka-angka di baris keempat menjadi 70, dan 70 mod 10 = 0. 4408 0412 3456 7893 adalah sebuah nomor kartu kredit yang sahih.

Sebagai contoh kedua, kita ambil angka 4417 1234 5678 9112

Major Industry Identifier (MII) adalah 4 (kategori bank dan finansial), *issuer identifier* adalah 441712 (partner dari VISA), nomor rekeningnya adalah 345678911, dan angka untuk mengecek adalah 2.

Mari kita coba pemeriksaan pada nomor 4417 1234 5678 9112, seperti pada contoh sebelumnya.

Kalau kita menjumlahkan semua angka pada baris terbawah, kita mendapatkan 69, lalu 69 mod 10 ≠ 0, sehingga kita menyimpulkan nomor 4417 1234 5678 9112 adalah nomor kartu kredit yang tidak sahih.

Dengan mengubah angka pengecekan dari 2 menjadi 3, kita mendapatkan angka baru 4417 1234 5678 9113, yang melewati pemeriksaan dengan algoritma Luhn, karena hasil penjumlahan baris terbawah menjadi 70, yang bisa dibagi 10. 4417 1234 5678 9113 adalah sebuah nomor kartu kredit yang sahih.

3.5. Implementasi algoritma Luhn dalam Bahasa C

Berikut ini diberikan contoh implementasi algoritma Luhn dalam bahasa C, sehingga setiap angka yang dimasukkan dapat langsung dicek apakah merupakan nomor kartu kredit yang sahih atau tidak.

Tabel 4: Contoh pertama

4	4	0	8	0	4	1	2	3	4	5	6	7	8	9	0
4 x 2 = 8	4	0 x 2 = 0	8	0 x 2 = 0	4	1 x 2 = 2	2	3 x 2 = 6	4	5 x 2 = 10	6	7 x 2 = 14	8	9 x 2 = 18	0
8	4	0	8	0	4	2	2	6	4	10 - 9 = 1	6	14 - 9 = 5	8	18 - 9 = 9	0
8	4	0	8	0	4	2	2	6	4	1	6	5	8	9	0

Tabel 5: Contoh kedua

4	4	1	7	1	2	3	4	5	6	7	8	9	1	1	2
4 x 2 = 8	4	1 x 2 = 2	7	1 x 2 = 2	2	3 x 2 = 6	4	5 x 2 = 10	6	7 x 2 = 14	8	9 x 2 = 18	1	1 x 2 = 2	2
8	4	2	7	2	2	6	4	10 - 9 = 1	6	14 - 9 = 5	8	18 - 9 = 9	1	2	2
8	4	2	7	2	2	6	4	1	6	5	8	9	1	2	2

```
//http://www.chriswareham.demon.co.uk/software/luhn.c

#include <ctype.h>
#include <stdio.h>
#include <string.h>


static int isValidNumber(const char *);

/*
 * Test harness for an implementation of the Luhn algorithm that checks the
 * validity of a credit card number.
 */
int
main(int argc, char *argv[])
{
   int i;

   if (argc < 2) {
      fprintf(stderr, "Usage: luhn <number>, ...\n");
      return 1;
   }

   for (i = 1; i < argc; ++i)
      printf("Number '%s' is%s a valid credit card number\n",
         argv[i], isValidNumber(argv[i]) ? "" : " not");

   return 0;
}


/*
 * Checks whether a string of digits is a valid credit card number according to
 * the Luhn algorithm.
 *
 * 1. Starting with the second to last digit and moving left, double the value
 *    of all the alternating digits. For any digits that thus become 10 or more,
 *    add their digits together. For example, 1111 becomes 2121, while 8763
 *    becomes 7733 (from (1+6)7(1+2)3).
 *
 * 2. Add all these digits together. For example, 1111 becomes 2121, then
 *    2+1+2+1 is 6; while 8763 becomes 7733, then 7+7+3+3 is 20.
 *
 * 3. If the total ends in 0 (put another way, if the total modulus 10 is 0),
 *    then the number is valid according to the Luhn formula, else it is not
 *    valid. So, 1111 is not valid (as shown above, it comes out to 6), while
 *    8763 is valid (as shown above, it comes out to 20).
 */

static int
isValidNumber(const char *number)
{
   int n, i, alternate, sum;

   if (!number)
      return 0;

   n = strlen(number);

   if (n < 13 || n > 19)
      return 0;

   for (alternate = 0, sum = 0, i = n - 1; i > -1; --i) {
      if (!isdigit(number[i]))
         return 0;

      n = number[i] - '0';

      if (alternate) {
         n *= 2;
         if (n > 9)
            n = (n % 10) + 1;
      }
      alternate = !alternate;

      sum += n;
   }

   return (sum % 10 == 0);
}
```

4. KESIMPULAN

Kesimpulan yang bisa diambil dari makalah ini adalah:

1. Nomor kartu kredit yang ada didasarkan pada algoritma Luhn sehingga bisa ditentukan dan dicari mana nomor kartu yang valid dan mana yang tidak.
2. Algoritma Luhn dengan dasar modulo 10 adalah salah satu bentuk implementasi dari Mata Kuliah Matematika Diskrit mengenai aritmetika modulo.
2. Transaksi online yang berbasiskan kartu kredit dengan hanya menggunakan validasi berupa nomor kartu, masa kadaluarsa kartu, nama pemilik masih tidak aman karena memungkinkan seseorang untuk menyalahgunakan nomor kartu orang lain.

DAFTAR REFERENSI

[1] Munir, Rinaldi. (2004). Bahan Kuliah IF5054 Kriptografi. Departemen Teknik Informatika, Institut Teknologi Bandung.

[2] Wikipedia (2007) Luhn Algorithm "http://en.wikipedia.org/wiki/Luhn_algorithm" Tanggal Akses : 1 Januari 2008-01-03

[3] Card Identification Features. 2007. http://www125.americanexpress.com/merchant/oam/resources/POS499.pdf". Tanggal akses : 1 Januari 2008.

[4] Gilleland, Michelle. 2007. Anatomy of Credit Card Numbers. "www.merriampark.com/anatomycc.htm". Tanggal akses : 1 Januari 2008