Aplikasi Teori Bilangan dalam Pembangkitan dan Validasi Nomor Kartu Kredit

Mohammad Taufan Tripurnasatria
13505121
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jalan Ganesha 10, Bandung 40132
Email: if15121@students.if.itb.ac.id

Abstrak *– Makalah ini membahas mengenai salah satu penggunaan teori bilangan, yang dalam hal ini memanfaatkan operasi aritmetika modulo serta sifat-sifatnya untuk melakukan pembangkitan dan validasi nomor kartu kredit. Pada makalah ini akan diberikan juga contoh kode program dalam bahasa C# untuk memperlihatkan bagaimana proses ini dilakukan dalam dunia nyata. Dalam melakukan validasi digunakan algoritma yang paling banyak digunakan untuk tujuan ini yaitu algoritma Luhn. Penggunaan validasi semacam merupakan hal yang sangat penting untuk diterapkan dalam transaksi di internet, terutama karena hingga saat ini keamanan transaksi menggunakan kartu kredit masih dipertanyakan. Namun perlu disadari bahwa algoritma ini ternyata masih memiliki banyak kelemahan, seperti pendeteksian pergeseran digit yang kurang cermat.*

Kata Kunci*: Kartu Kredit, Validasi, Luhn Algorithm, Aritmetika Modulo*

1. PENDAHULUAN

Meningkatnya kebutuhan untuk melakukan transaksi *online* dalam dekade terakhir ini mendorong pertumbuhan aplikasi web yang mampu melayani transaksi yang cepat dan murah. Salah satu solusinya adalah penggunaan kartu kredit. Namun, dilihat dari segi keamanannya, keamanan kartu kredit sangat rapuh. Dalam melakukan transaksi pembayaran *online*, seseorang bisa saja memasukkan nomor palsu (tidak valid) ke dalam form transaksi, untuk itu perlu dilakukan validasi terlebih dahulu yang bisa langsung diterapkan pada aplikasi *web* untuk mengecek apakah nomor yang diajukan oleh pelanggan valid atau tidak sebelum dilakukan *authorization* (pengesahan pembayaran) lebih lanjut.

Untuk melakukan validasi ini, tidak diperlukan algoritma validasi yang rumit. Cukup dengan memanfaatkan salah satu algoritma yang sangat populer untuk validasi nomor kartu, yaitu algoritma Luhn. Algoritma ini terbilang sangat sederhana karena hanya memanfaatkan sifat kongruen dari operasi modulo (dalam hal ini modulo 10), namun cukup ampuh untuk menjamin nomor kartu tersebut sudah valid atau tidak.

2. KARTU KREDIT DAN KEAMANANNYA

2.1 Pengertian

Kartu kredit adalah sistem pembayaran di mana penerbit (*issuer*) kartu meminjamkan uang kepada pelanggan (*cardholder*) untuk dibayarkan ke penerima pembayaran transaksi dengan kartu kredit (*merchant*). Sebuah kartu kredit berbeda dari kartu debit, karena kartu kredit tidak menghapus/mengurangi uang dari rekening pelanggan setelah transaksi. Hampir semua kartu kredit memiliki bentuk dan ukuran yang sama, sesuai dengan spesifikasi standar ISO 7810[5].

2.2 Pihak yang terlibat

Dalam penerbitan dan transaksi menggunakan kartu kredit terdapat beberapa pihak yang terlibat antara lain[6]:

a. Pemegang kartu : Pemilik kartu yang melakukan transaksi menggunakan kartu kredit (pelanggan).
b. Bank penerbit kartu : lembaga keuangan atau organisasi lain yang mengeluarkan/menerbitkan kartu kredit ke pemegang kartu.
c. *Merchant* : individu atau badan usaha yang menerima pembayaran lewat kartu kredit untuk produk dan jasa yang dijual ke pemegang kartu.
d. *Acquiring bank* : lembaga keuangan/organisasi yang yang mengumpulkan permintaan pengesahan kredit (*credit authorization request*) dari *merchant* dan menyediakan jaminan pembayaran.

Beberapa bank penerbit kartu juga merupakan *acquiring bank*. Beberapa bank penerbit kartu mengizinkan *acquiring bank* lain untuk melakukan pengesahan (*authorization*) untuk mereka.

2.3 Keamanan kartu kredit

Keamanan kartu kredit tergantung pada kerahasiaan nomor kartunya. Ini berarti bahwa ketika seseorang (selain pemegang kartu) membaca nomor kartu, keamanannya akan menjadi lemah. Karena hal ini terjadi hampir sepanjang waktu yaitu ketika transaksi dilakukan, maka dapat dikatakan keamanan kartu

kredit secara umum sangat lemah. Namun, seorang *user* yang hanya mengetahui nomor kartu kredit saja hanya bisa melakukan transaksi tertentu. *Merchant* biasanya akan menerima nomor kartu kredit tanpa verifikasi tambahan untuk pengiriman barang yang dipesan, namun alamat pengiriman akan dicatat, jadi penjahat kartu kredit harus memberikan alamat *anonymous* (bukan miliknya) dan mengumpulkan barang hasil pesanannya tersebut tanpa diketahui. Beberapa *merchant* akan menerima pembayaran dengan memberikan nomor kartu kredit hanya untuk pembelian di tempat (toko) karena akses terhadap nomor kartu seringkali memudahkan kejahatan, tetapi banyak yang mensyaratkan kartunya sendiri diperlihatkan, dan harus memberikan tanda tangan.
Kartu yang dicuri dapat dibatalkan (di-*block*), dan jika hal ini dilakukan dengan sangat cepat, tidak ada perampokan yang dapat berlangsung terhadap kartu tersebut. Untuk pembelian melalui internet, level keamanannya sama dengan lewat pengiriman (hanya mensyaratkan kita mengetahui nomor kartu) sehingga pencuri hanya perlu memikirkan bagaimana mengumpulkan barang hasil pesanan tanpa diketahui. Salah satu teknik yang paling aman untuk mencegah hal ini adalah dengan mensyaratkan PIN keamanan pada kartu (seperti pada kartu telepon selular) yang tentu saja harus menyertakan kartunya.

2.4 Penomoran kartu kredit

Karena keamanan kartu kredit sangat bergantung pada keamanan nomor kartunya, maka hal ini sangat penting untuk dijelaskan. Nomor yang terdapat pada kartu kredit memiliki beberapa struktur internal dan semua kartu memiliki kesamaan skema penomoran.
Berikut penjelasan mengenai skema penomoran kartu kredit[1]:

a. *Major Industry Identifier* (MII)
Digit pertama pada nomor kartu kredit adalah MII yang mewakili kategori lembaga yang menerbitkan kartu kredit. Digit MII mewakili kategori penerbit (*issuer*) berikut:

Tabel 1 Kategori MII[1]

Digit MII	Kategori Penerbit
0	*ISO/TC 68 and other industry assignments*
1	*Airlines*
2	*Airlines and other industry assignments*
3	*Travel and entertainment*
4	*Banking and financial*
5	*Banking and financial*
6	*Merchandizing and banking*
7	*Petroleum*
8	*Telecommunication and other industry assignments*
9	*National assignment*

Sebagai contoh, American Express, Diner's Club, dan Carte Blanche berada di kategori 3 (*travel and entainment*), VISA, MasterCard, dan Discover berada di kategori 4 dan 5 (*banking and financial*), dan SUN Oil dan Exxon berada di kategori 7 (*petroleum*).

b. *Issuer Identifier* (II)
Enam digit pertama pada nomor kartu kredit (termasuk digit MII) membentuk *issuer identifier*, yaitu nomor penerbit. Ini berarti jumlah penerbit kartu kredit yang mungkin adalah 10^6 atau 1.000.000 penerbit. Beberapa penerbit kartu kredit yang terkenal dan umum digunakan adalah sebagai berikut:

Tabel 2 Kategori II[1]

Penerbit	Identifier	Panjang nomor kartu
Diner's Club/Carte Blanche	300xxx-305xxx, 36xxxx, 38xxxx	14
American Express	34xxxx, 37xxxx	15
VISA	4xxxxx	13, 16
MasterCard	51xxxx-55xxxx	16
Discover	6011xx	16

c. Nomor Rekening (*Account Number*)
Digit ke 7 (tujuh) hingga ke (n – 1) dari nomor kartu kredit adalah nomor rekening yang unik. Panajang maksimum dari nomor kartu kredit adalah 19 digit. Karena 6 (enam) digit pertama merupakan II dan digit terakhir merupakan *check digit*, berarti panjang maksimum dari nomor rekening adalah 12 digit. Jadi, terdapat 10^{12} atau 1.000.000.000.000 kemungkinan nomor rekening.

d. *Check Digit*
Digit terakhir dari nomor kartu kredit adalah *check digit*. Digit ini digunakan untuk memeriksa validitas nomor kartu. Algoritma yang paling sering digunakan untuk memeriksa nomor kartu kredit adalah algoritma Luhn.

3. ARITMETIKA MODULO

Aritmetika modulo (*modular arithmetic*) memainkan peranan yang penting dalam komputasi integer, khususnya pada aplikasi kriptografi. Operator yang digunakan pada aritmetika modulo adalah **mod**. Operator mod memberikan sisa pembagian. Misalnya 23 dibagi 5 memberikan hasil = 4 dan sisa = 3, sehingga kita tulis 23 mod 5 = 3. Definisi dari operator mod dinyatakan sebagai berikut[4]:

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* disebut modulus atau modulo, dan hasil artimetika modulo *m* terletak di dalam himpunan {0, 1, 2, ..., *m* - 1}.

3.1 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*) .....(1)
(notasi '≡' dibaca 'kongruen')
Jika *a* tidak kongruen dengan *b* dalam modulus *m*, maka ditulis

$a \not\equiv b$ (**mod** *m*) .....(2)

Misalkan 39 **mod** 5 = 4 dan 24 **mod** 5 = 4, maka 39 ≡ 24 (**mod** 5). Definisi formal dari kongruen adalah sebagai berikut[4]:

Misalkan *a* dan *b* adalah bilangan bulat dan *m* adalah bilangan > 0, maka *a* ≡ *b* (**mod** *m*) jika *m* habis membagi *a* – *b*.

4. ALGORITMA LUHN

Algoritma Luhn (sering juga disebut Formula Luhn, atau algoritma mod 10) adalah formula sederhana untuk memvalidasi berbagai macam nomor kartu, misalnya kartu kredit. Formula ini diciptakan oleh seorang ilmuwan dari IBM bernama Hans Peter Luhn. Algoritma ini merupakan *public domain* dan sangat banyak digunakan pada saat ini. Algoritma ini tidak ditujukan sebagai fungsi *hash* yang aman secara kriptografis; algoritma ini dirancang untuk melindungi terhadap kesalahan teknis, bukan serangan yang berbahaya. Sebagian besar kartu kredit dan berbagai nomor identifikasi yang dikeluarkan pemerintah di berbagai negara menggunakan algoritma ini sebagai metode sederhana untuk membedakan nomor yang valid dari berbagai digit random.
Langkah-langkah berikut dapat menggambarkan secara rinci tentang algoritma ini[3]:

Pertama: kalikan dengan 2 (dua) setiap digit secara berselang-seling nomor kartu selain *check digit* dimulai dari dua digit dari kanan(*check digit* dihitung yang pertama)
Kedua : jika ada hasil penjumlahan yang lebih dari 9, kurangkan dengan 9
Ketiga : jumlahkan seluruh digit dari MII hingga *check digit*, jika diperoleh bilangan yang kongruen dengan 10 (**mod** 10) maka nomor kartu kredit tersebut valid, jika tidak maka tidak valid.

5. IMPLEMENTASI ALGORITMA LUHN UNTUK PEMBANGKITAN NOMOR KARTU KREDIT

Berikut adalah potongan kode program dalam bahasa C# untuk membangkitkan nomor kartu kredit yang memenuhi kriteria Luhn[5]:

```
int[] CreateNumber(int length)
 {
   Random random = new Random();
   int[] digits = new int[length];
   // For loop keeps default value of zero for
last slot in array
   for(int i = 0; i < length - 1; i++)
   {
     digits[i] = random.Next(10);
   }
   int sum = 0;
   bool alt = true;
   for(int i = length - 2; i >= 0; i--)
   {
     if(alt)
     {
       int temp = digits[i];
       temp *= 2;
       if(temp > 9)
       {
         temp -= 9;
       }
       sum += temp;
     }
     else
     {
       sum += digits[i];
     }
     alt = !alt;
   }
   int modulo = sum % 10;
   if(modulo > 0)
   {
     digits[length-1] = 10 - modulo;
   }
   // No else req'd - keep default value of
zero for digits[length-1]
   return digits;
 }
```

6. IMPLEMENTASI ALGORITMA LUHN UNTUK MEMVALIDASI NOMOR KARTU KREDIT

Berikut adalah potongan kode dalam bahasa C# untuk memvalidasi nomor kartu kredit[5]:

```
bool CheckNumber(int[] digits)
 {
   int sum = 0;
   bool alt = false;
   for(int i = digits.Length - 1; i >= 0;
i--)
   {
     if(alt)
     {
       digits[i] *= 2;
       if(digits[i] > 9)
       {
         digits[i] -= 9;
       }
     }
     sum += digits[i];
     alt = !alt;
   }
   return sum % 10 == 0;
 }
```

7. HASIL DAN PEMBAHASAN

Berikut ini contoh hasil eksekusi implementasi algoritma validasi di atas (setelah ditambahkan program utama):

```
> luhnval 378282246310005
> Number '378282246310005' is valid
> luhnval 6011111111111117
> Number '6011111111111117' is valid
> luhnval 6011111111111116
> Number '6011111111111116' is not valid
> luhnval 4222222222222
> Number '4222222222222' is valid
> luhnval 422222222222
> Number '422222222222' is not valid
> luhnval 1111111118901
> Number '1111111118901' is valid
> luhnval 1111111118091
> Number '1111111118091' is valid
> luhnval 0000000008904
> Number '0000000008904' is valid
> luhnval 00000000008904
> Number '00000000008904' is valid
> luhnval 000000000008904
> Number '000000000008904' is valid
```

Berikut ini pembahasan beberapa contoh kasus di atas untuk diperlihatkan bagaimana algoritma Luhn ini dieksekusi.

Nomor kartu : **378282246310005**
Major Industry Identifier dari nomor ini adalah 3 (*Travel and Entertainment*), *issuer identifier* 378282, nomor rekening 24631000, dan *check digit*-nya adalah 5.

Jika kita terapkan algoritma Luhn pada nomor di atas diperoleh tabel berikut:

3	7	8	2	8	2	2	4	6	3	1	0	0	0	5
3	14	8	4	8	4	2	8	6	6	1	0	0	0	5
3	5	8	4	8	4	2	8	6	6	1	0	0	0	5

- Baris pertama adalah nomor yang sebenarnya.
- Baris kedua adalah perkalian dua dari digit genap yang dimulai dari kanan.
- Baris ketiga adalah hasil normalisasi, yaitu jika digit pada langkah kedua nilainya lebih besar dari 9 maka dikurangi dengan 9.
- Jumlah dari semua angka pada baris ketiga adalah 60 ≡ 10 (**mod** 10), maka nomor ini valid.

Nomor kartu: **1111111118901**

1	1	1	1	1	1	1	1	1	8	9	0	1
1	2	1	2	1	2	1	2	1	16	9	0	1
1	2	1	2	1	2	1	2	1	7	9	0	1

- Baris pertama adalah nomor yang sebenarnya
- Baris kedua adalah perkalian dua dari digit genap yang dimulai dari kanan
- Baris ketiga adalah hasil normalisasi, yaitu jika digit pada langkah kedua nilainya lebih besar dari 9 maka dikurangi dengan 9.
- Jumlah dari semua angka pada baris ketiga adalah 30 ≡ 10 (**mod** 10), maka nomor ini valid.

Nomor kartu: **1111111118091**

1	1	1	1	1	1	1	1	1	8	0	9	1
1	2	1	2	1	2	1	2	1	16	0	18	1
1	2	1	2	1	2	1	2	1	7	0	9	1

- Baris pertama adalah nomor yang sebenarnya
- Baris kedua adalah perkalian dua dari digit genap yang dimulai dari kanan
- Baris ketiga adalah hasil normalisasi, yaitu jika digit pada langkah kedua nilainya lebih besar dari 9 maka dikurangi dengan 9.
- Jumlah dari semua angka pada baris ketiga adalah 30 ≡ 10 (**mod** 10), maka nomor ini valid.

Dua contoh terakhir di atas menunjukkan salah satu kelemahan algoritma ini, yaitu tidak mampu mendeteksi pertukaran dua digit, 90 ke 09 dan 09 ke 90.

Nomor kartu: **0000000008904**

0	0	0	0	0	0	0	0	0	8	9	0	4
0	0	0	0	0	0	0	0	0	16	9	0	4
0	0	0	0	0	0	0	0	0	7	9	0	4

- Baris pertama adalah nomor yang sebenarnya
- Baris kedua adalah perkalian dua dari digit genap yang dimulai dari kanan

- Baris ketiga adalah hasil normalisasi, yaitu jika digit pada langkah kedua nilainya lebih besar dari 9 maka dikurangi dengan 9.
- Jumlah dari semua angka pada baris ketiga adalah 20 ≡ 10 (**mod** 10), maka nomor ini valid.

Nomor kartu: **00000000008904**

0	0	0	0	0	0	0	0	0	0	8	9	0	4
0	0	0	0	0	0	0	0	0	0	16	9	0	4
0	0	0	0	0	0	0	0	0	0	7	9	0	4

- Baris pertama adalah nomor yang sebenarnya
- Baris kedua adalah perkalian dua dari digit genap yang dimulai dari kanan
- Baris ketiga adalah hasil normalisasi, yaitu jika digit pada langkah kedua nilainya lebih besar dari 9 maka dikurangi dengan 9.
- Jumlah dari semua angka pada baris ketiga adalah 20 ≡ 10 (**mod** 10), maka nomor ini valid.

Dua contoh terakhir di atas menunjukkan kelemahan yang lain dari algoritma ini yaitu tidak bisa mendeteksi kelebihan angka 0 (nol) di bagian kiri nomor. Hal ini disebabkan perkalian dengan dua dimulai dari kiri sehingga penambahan di kanan tidak akan mempengaruhi hasil akhirnya.

8. KESIMPULAN

Dari hasil dan pembahasan yang telah diperoleh di atas dapat disimpulkan beberapa hal sebagai berikut:

a. Algoritma Luhn cukup ampuh dalam memvalidasi nomor kartu kredit karena dapat mendeteksi kesalahan yang tidak disengaja, misalnya kesalahan entri oleh *user* dengan menambahkan digit yang sama dengan angka pada digit sebelumnya seperti pada contoh di atas.

b. Algoritma Luhn dapat mendeteksi beberapa kesalahan akibat pergeseran posisi digit pada nomor yang dimasukkan.

c. Algoritma Luhn dapat digunakan sebagai langkah preventif awal dalam mencegah tindak kriminal karena kebocoran nomor kartu kredit.

d. Terdapat setidaknya 2 (dua) kelemahan algoritma ini:
   1. Pergeseran posisi 09 ke 90 tidak dapat dideteksi, begitu juga sebaliknya dari posisi 90 ke 09 tidak dapat dideteksi. Keduanya akan memberikan hasil yang sama ketika diperiksa dengan algoritma ini.
   2. Penambahan angka 0 (nol) di bagian depan nomor kartu kredit tidak mempengaruhi hasil, sehingga dapat mengecoh, seperti pada contoh di atas. Hal ini disebabkan oleh perkalian dua dimulai dari kanan sehingga hasilnya akan sama jika ditambahkan nol di bagian kiri, karena tidak akan mempengaruhi hasil jika dikalikan dua.

e. Kekurangan dari algoritma Luhn di atas dapat diperbaiki dengan menerapkan algoritma lain yang lebih kompleks, seperti algoritma Verhoeff yang dapat mendeteksi kesalahan pergeseran ataupun pertukaran digit.

f. Algoritma Luhn terbukti telah berhasil menunjukkan bahwa teori matematika diskrit (misalnya teori bilangan bulat) dapat diaplikasikan di dunia nyata untuk meningkatkan keamanan dan kenyamanan hidup manusia.

DAFTAR REFERENSI

[1] Gilleland, Michael, *Anatomy of Credit Card Numbers*, http://euro.ecom.cmu.edu/resources/elibrary/everycc.htm , tanggal akses: 31 Desember 2007, pukul 20.00 WIB

[2] Golbguru, *How to Generate *Valid* Credit Card Numbers*, http://www.thetaoofmakingmoney.com/2007/04/12/324.html , tanggal akses: 01 Januari 2008, pukul 07.00 WIB

[3] Hstiles, *Credit Card Validation-Check Digits*, http://www.beachnet.com/~hstiles/cardtype.html , tanggal akses: 01 Januari 2008, pukul 07.30 WIB

[4] Munir, Rinaldi, Matematika Diskrit, Penerbit Informatika, 2005

[5] Wikipedia, *Wikipedia - The Free Encyclopedia*, http://en.wikipedia.org/ , tanggal akses: 01 Januari 2008, pukul 06.50 WIB

[6] Ziegler, Joe, *Everything you ever wanted to know about CC's*, http://euro.ecom.cmu.edu/resources/elibrary/everycc.htm , tanggal akses: 01 Januari 2008, pukul 07.10 WIB