Pembangkitan Nomor Kartu Kredit dan Pengecekannya Dengan Menggunakan Algoritma Luhn

Shanny Avelina Halim (13504027)
Program Studi Teknik Informatika Institut Teknologi Bandung
email: if14027@Students.if.itb.ac.id

Abstrak – Di dalam makalah ini, topik yang akan dibahas adalah mengenai kriptografi yang diterapkan di dalam kehidupan sehari-hari. Salah satunya adalah pembangkitan nomor untuk kartu kredit yang baik sehingga saat terjadi kesalahan pengetikan dapat segera diketahui. Selain itu juga akan dibahas mengenai cara pengecekan nomor katu kredit yang dimasukkan dengan menggunakan algoritma Luhn untuk menentukan apakah nomor kartu kredit tersebut merupakan nomor yang valid atau bukan. Untuk mendukung pembuatan makalah ini, juga dibuat sebuah simulasi untuk membangkitkan nomor kartu kredit tersebut yang terangkum dalam file gen.html dan cek.html.

Kata Kunci: kartu kredit, pembangkitan, pengecekan, acak, Luhn.

1. PENDAHULUAN

Salah satu jenis kriptografi yang dapat diterapkan dalam kehidupan sehari-hari adalah cara membangkitkan bilangan-bilangan untuk identitas sebuah barang atau orang yang unik dan baik, sehingga bila terjadi kesalahan pengetikan, dapat diketahui dan diantisipasi. Contohnya adalah nomor barang yang dijual di sebuah supermarket dan nomor kartu kredit. Namun pada makalah ini contoh yang akan ditelaah adalah pembangkitan nomor kartu kredit yang baik dan pengecekannya dengan menggunakan alghoritma Luhn.

2. PEMBANGKITAN NOMOR KARTU KREDIT

2.1 Nomor kartu kredit

Pada sistem yang terdapat pada kasir-kasir supermarket, bila nomor identitas setiap barang yang ada dibuat berurutan, maka saat kasir salah memasukkan satu selisih angka saja, sistem akan tetap menganggap nomor yang dimasukkan tersebut valid walaupun jenis barang yang diinginkan oleh pembeli tidak sama dengan jenis barang yang dimasukkan datanya oleh kasir. Hal yang sama berlaku terhadap nomor kartu kredit. Bila nomor kartu kredit dibuat berurutan, maka sistem akan menganggap setiap nomor yang dimasukkan saat melakukan transaksi hampir selalu merupakan nomor valid.

Berdasarkan masalah tersebut di atas, maka dapat dilakukan cara baru untuk membangkitkan nomor kartu kredit yang baik. Nomor kartu kredit sekarang ini merupakan hasil operasi dari nomor identitas yang dimiliki oleh tiap-tiap penyelenggara kartu kredit. Nomor kartu kredit pada umumnya terdiri dari 13 sampai 16 digit. Untuk mencegah nomor kartu kredit yang berurutan, biasanya nomor-nomor yang dimiliki olem masing-masing pengguna kartu kredit merupakan hasil operasi dari angka-angka yang disebut Prefix dan CheckDigit.

Pada makalah ini, semua penghitungan digit selalu dimulai dari kiri. Digit 1 adalah angka yang terletak paling kiri dan digit terakhir adalah digit yang terletak paling kanan.

Prefix terletak pada satu atau lebih digit pertama nomor katu kredit. Prefix merupakan identitas yang dimiliki oleh penyelenggara kartu kredit sehingga setiap penyelenggara memiliki Prefix masing-masing yang berbeda.

CheckDigit terletak pada satu digit terakhir pada rangkaian nomor kartu kredit. CheckDigit adalah bilangan yang ditambahkan untuk memastikan nomor kartu kredit akan dinyatakan valid saat dilakukan pengecekan. Contoh Prefix yang banyak diketahui adalah:

VISA	: 4
MasterCard	: 51 - 55
American Express	: 34, 37

Panjang nomor kartu kredit pun berbeda-beda tergantung dari penyelenggaranya. Kebanyakan nomor kartu kredit berkisar antara 13 sampai 16 digit, namun ada pula yang mencapai 19 digit. Berikut ini adalah panjang nomor kartu kredit dari 3 bank yang telah disebutkan:

VISA	: 16 digit
MasterCard	: 15 digit
American Express	: 16 digit

2.2 Algoritma Luhn

Algoritma Luhn diciptakan pertama kali oleh seorang peneliti dari IBM, yaitu Hans Peter Luhn dan algoritma tersebut diresmikan atas namanya pada

tanggal 23 Agustus 1960. Algoritma ini memiliki tujuan untuk mengantisipasi kesalahan pengetikan pada saat memasukkan informasi yang dibutuhkan. Algoritma Luhn juga biasa disebut algoritma "modulus 10" atau "mod 10" karena dalam proses pengecekan validitas nomor kartu kredit dilakukan pembagian dengan angka 10. Hasil baginya akan menentukan apakah nomor kartu kredit yang dicek valid atau tidak.

Algoritma Luhn hanya dapat menangani kesalahan pengetikan yang terjadi pada satu digit angka saja, seperti 1111 menjadi 1112. Untuk menangani kesalahan pengetikan yang jumlahnya lebih dari 1 digit, seperti 1111 menjadi 1133, dibutuhkan algoritma pengecekan lain yang lebih kompleks dari algoritma Luhn seperti algoritma Verhoeff. Namun pada makalah ini hanya akan dibahas mengenai algoritma Luhn.

Langkah-langkah pengecekan nomor kartu kredit dengan menggunakan algoritma Luhn beserta contoh penerapannyanya akan dijelaskan dengan lebih rinci pada subbab 2.4

2.3 Langkah-langkah pembangkitan

Pembangkitan nomor yang dilakukan oleh penyelenggara kartu kredit merupakan sebuah proses yang kompleks dan bukan merupakan hal yang diketahui oleh kalangan publik sehingga pembangkitan nomor-nomor kartu kredit rekaan yang dilakukan oleh penulis merupakan pembangkiran yang berdasarkan pada pepembalikan proses dari algoritma Luhn.

Berikut ini adalah salah satu contoh nomor kartu kredit yang dikeluarkan oleh VISA.

4247933049116517

Nomor kartu kredit VISA terdiri dari 16 angka. Prefix dari nomor tersebut terletak pada digit satu, yaitu 4. CheckDigit dari nomor tersebut terletak pada digit 16, yaitu 7. Digit 2 sampai digit 16 dibangkitkan dari Prefix.

Mula-mula akan dibangkitkan bilangan 14 buah bilangan acak untuk mengisi digit 2 sampai 15. Dengan asumsi bahwa nomor kartu kredit di atas belum dibangkitkan, maka nilai CheckDigit pada digit 16 belum diketahui dan kita anggap bahwa bilangan acak yang dibangkitkan untuk mengisi digit 2 sampai 15 adalah:

24793304911651

Sehingga bila nomor kartu kredit tersebut dipecah, maka komposisi dari angka-angka penyusunnya adalah sebagai berikut:

4 24793304911651 _

Selanjutnya yang perlu dilakukan adalah menentukan sebuah angka untuk CheckDigit sehingga nomor kartu kredit di atas menjadi sebuah nomor yang valid saat dilakukan pengecekan dengan algoritma Luhn. Satu buah angka tersebut akan ditentukan berdasarkan operasi yang dilakukan terhadap 15 buah angka yang telah diketahui.

Untuk menentukan nilai CheckDigit, ada beberapa langkah yang perlu dilakukan, antara lain:

1. Mengalikan semua bilangan yang berada pada digit ganjil dengan bilangan 2. Bila nilainya lebih dari 9, maka dikurangi dengan 9.
2. Menjumlahkan semua hasil perkalian pada digit ganjil di atas.
3. Menjumlahkan semua bilangan yang berada pada digit genap.
4. Jumlahkan nilai pada langkah 2 dan 3.
5. Nilai CheckDigit adalah 10 – (nilai akhir modulus 10)

Berikut ini merupakan langkah-langkah yang diterapkan terhadap 15 digit yang sudah diketahui dari contoh nomor kartu kredit.

1. Langkah 1
   (Digit 1) * 2 = 4 * 2 = 8
   (Digit 3) * 2 = 4 * 2 = 8
   (Digit 5) * 2 = 9 * 2 = 18 – 9 = 9
   (Digit 7) * 2 = 3 * 2 = 6
   (Digit 9) * 2 = 4 * 2 = 8
   (Digit 11) * 2 = 1 * 2 = 2
   (Digit 13) * 2 = 6 * 2 = 12 – 9 = 3
   (Digit 15) * 2 = 1 * 2 = 2

2. Langkah 2
   8 + 8 + 9 + 6 + 8 + 2 + 3 + 2 = 46

3. Langkah 3
   2 + 7 + 3 + 0 + 9 + 1 + 5 = 27

4. Langkah 4
   46 + 27 = 73

5. Langkah 5
   CheckDigit = 10 – (73 % 10) = 10-3 = 7

Perlu diingat bahwa langkah tersebut di atas hanya dapat dilakukan bila jumlah seluruh digit yang diinginkan jumlahnya genap, seperti milik VISA, yaitu 16. Untuk membangkitkan nomor kartu kredit dengan jumlah digit ganjil, seperti 13 digit atau 15 digit, hal yang perlu dilakukan adalah menukar penerapannya pada digit ganjil dan genap, yaitu angka-angka pada digit genap yang dikali dengan 2,

dan angka-angka pada digit ganjil yang langsung dijumlahkan.

Setelah melakukan langkah-langkah di atas, maka didapat nilai CheckDigit, yaitu 7. Selanjutnya yang perlu dilakukan adalah merangkaikan Prefix, bilangan acak, dan CheckDigit menjadi sebuah nomor kartu kredit, yaitu:

4247933049116517

Kartu kredit dengan nomor seperti tertera di atas, hasilnya akan valid bila dicek dengan menggunakan algoritma Luhn.

2.4 Pengecekan nomor kartu kredit

Langkah-langkah yang dilakukan untuk mengecek validitas dari sebuah nomor kartu kredit dengan menggunakan algoritma Luhn tidak jauh berbeda dengan langkah-langkah untuk menemukan nilai CheckDigit. Untuk mencari nilai CheckDigit pada kartu kredit dari VISA, maka operasi dilakukan terhadap 15 dari 16 digit yang telah diketahui, sedangkan untuk pengecekan dengan algoritma Luhn, operasi dilakukan terhadap seluruh nomor yang diketahui, yaitu 16 digit. Langkah-langkah yang dilakukan untuk pengecekan dengan algoritma Luhn antara lain:

1. Mengalikan semua bilangan yang berada pada digit ganjil dengan bilangan 2. Bila nilainya lebih dari 9, maka dikurangi dengan 9.
2. Menjumlahkan semua hasil perkalian pada digit ganjil di atas.
3. Menjumlahkan semua bilangan yang berada pada digit genap.
4. Jumlahkan nilai pada langkah 2 dan 3.
5. Bagi hasil pada langkah 4 dengan 10. Bila hasil akhir habis dibagi 10, maka nomor kartu kredit tersebut valid. Bila hasil akhir tidak habis dibagi 10, maka nilai kartu kredit tersebut tidak valid.

Pada contoh katu kredit VISA dengan nomor 4247933049116517, langkah-langkah yang dilakukan untuk pengecekan validitas kartu adalah sebagai berikut:

1. Langkah 1
   (Digit 1) * 2 = 4 * 2 = 8
   (Digit 3) * 2 = 4 * 2 = 8
   (Digit 5) * 2 = 9 * 2 = 18 – 9 = 9
   (Digit 7) * 2 = 3 * 2 = 6
   (Digit 9) * 2 = 4 * 2 = 8
   (Digit 11) * 2 = 1 * 2 = 2
   (Digit 13) * 2 = 6 * 2 = 12 – 9 = 3
   (Digit 15) * 2 = 1 * 2 = 2
2. Langkah 2
   8 + 8 + 9 + 6 + 8 + 2 + 3 + 2 = 46
3. Langkah 3
   2 + 7 + 3 + 0 + 9 + 1 + 5 + 7 = 34
4. Langkah 4
   46 + 34 = 80
5. Langkah 5
   Karena 80 habis dibagi 10, maka nilai kartu kredit tersebut valid

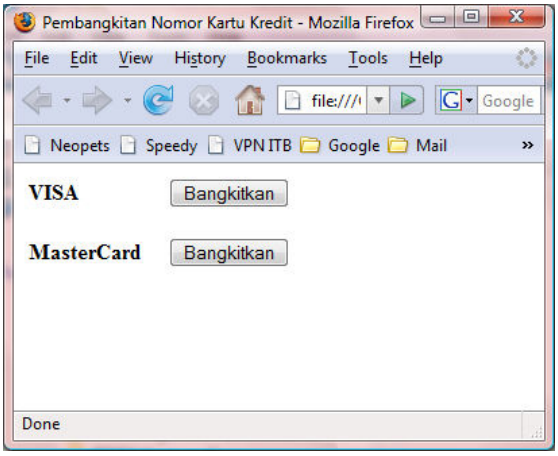
Sama seperti pembangkitan bilangan untuk mendapatkan nilai CheckDigit, langkah-langkah di atas juga hanya dapat diterapkan untuk kartu kredit yang mamiliki panjang digit genap, seperti 16. Untuk kartu kredit dengan panjang digit ganjil, seperti 13 digit atau 15 digit, maka penerapannya pada digit ganjil dan genap ditukar, angka-angka pada digit genap yang dikali 2 dan angka-angka pada digit ganjil yang langsung dijumlahkan

3. HASIL DAN PEMBAHASAN

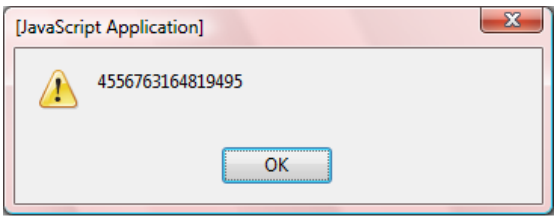
Sebagai hasil dari pembahasan yang dilakukan oleh penulis, maka telah membuat sebuah simulasi yang dapat membangkitkan nomor kartu kredit dari VISA dan MasterCard. Perlu diingat bahwa nomor-nomor kartu kredit yang dihasilkan adalah nomor-nomor yang akan melewari pengecekan dengan algoritma Luhn dan hanya sedikit kemungkinannya bahwa nomor-nomor tersebut dapat digunakan untuk melakukan transaksi pembayaran melalui internet. Simulasi tersebut terdiri dari sebuah halamanan web dan dapat dijalankan melalui browser Mozilla Firefox. Untuk membuat fungsi-fungsi yang diperlukan, penulis menggunakan bahasa pemrograman Javascript.

Dua buah file tersebut adalah gen.html untuk membangkitkan nomor kartu kredit, dan cek.html untuk mengecek sebuah nomor kartu kredit. Berikut ini akan dilakukan pembahasan dengan lebih rinci mengenai fungsi-fungsi yang ada dalam file gen.html dan cek.html

3.1 gen.html



Gambar di atas adalah gambar tampilan muka dari file gen.html.



Gambar di atas adalah tampilan saat tombol untuk membangkitkan nomot kartu kredit dari VISA ditekan oleh pengguna.

Berikut ini akan dijelaskan mengenai kode program dari file gen.html dengan lebih rinci:

1. Mula-mula dibuat dua buah array yang isinya merupakan beberapa buah prefix dari penyelenggara kartu kredit. Elemen array tersebut berupa string, dan bukan integer. Hal ini bertujuan untuk memperkecil jumlah operasi yang dilakukan untuk membangkitkan bilangan acak. Pengubahan bentuk dari string menjadi integer hanya dilakukan untuk menerapkan operasi matematika.

   `visaPrefixList` (VISA)
   `mastercardPrefixList` (MasterCard)

2. Fungsi Fungsi `strrev()`
   Parameter dari fungsi `strrev` adalah satu buah string yang terdiri dari simbol numerik. Fungsi ini digunakan untuk posisi pembacaan kartu kredit. Bila mula-mula urutan nomor dari digit paling kiri sampai digit paling kanan, maka akan diubah urutannya dari digit paling kanan sampai digiit paling kiri. Fungsi ini diperlukan untuk mengantisipasi perbedaan nomor kartu kredit yang memiliki panjang nomor ganjil dan genap sehingga langkah-langkah pembangkitan dan pengecekan pada subbab 2.3 dan 2.4 dapat dilakukan dengan benar.

3. Fungsi `credit_card_number()`
   Fungsi ini digunakan untuk memilih Prefix secara acak dari daftar Prefix yang dimiliki oleh tiap-tiap penyelenggara kartu kredit. Parameternya adalah penyelenggara kartu kredit yang dipilih oleh pengguna, yaitu VISA, atau MasterCard, panjang nomor kartu kredit yang diinginkan, dan banyaknya nomor kartu kredit yang diinginkan untuk dibangkitkan.

4. Fungsi `completed_number()`
   Fungsi ini adalah fungsi yang digunakan untuk menciptakan nomor kartu kredit yang valid dan lulus dari pengecekan dengan menggunakan algoritma Luhn. Parameternya adalah Prefix yang telah dipilih secara acak dari fungsi `credit_card_number` di atas, dan panjang nomor kartu kredit yang diinginkan.

   Mula-mula dibangkitkan bilangan acak dari digit setelah Prefix sampai digit sebelum CheckDigit. Karena Prefix bentuknya adalah string, maka untuk lebih mempersingkat waktu, bilangan acak yang dibangkitkan dan kemudian ditambahkan juga bentuknya adalah string.

   Misalnya bilangan yang dibangkitkan adalah 2, 1583, dan 42, maka bilangan acak yang dihasilkan adalah 2158342 dan bukan hasil penambahan ketiga bilangan tersebut, yaitu 1627. Selanjutnya Prefix dan bilangan acak juga dirangkaikan.

   Selanjutnya nomor kartu kredit tersebut dibalik urutannya dengan menggunakan fungsi `strrev` yang sudah disebutkan di atas. Langkah yang perlu dilakukan selanjutnya adalah mencari nilai dari CheckDigit agar nomor kartu lulus dari pengecekan algoritma Luhn. Hal tersebut dilakukan dengan mengubah langkah-langkah pada subbab 2.3 ke dalam bentuk operasi matematika. Nomor kartu kredit sementara juga diubah bentuknya dari string menjadi integer sehingga operasi matematika dapat diterapkan.

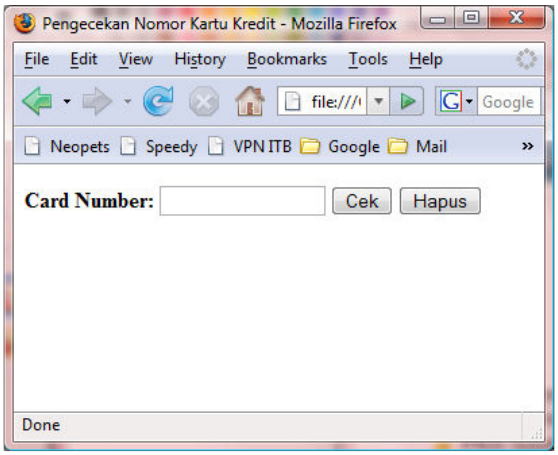
   Setelah nilai CheckDigit didapat, hal yang perlu dilakukan adalah merangkaikannya dengan nomor kartu kredit sementara yang telah diketahui sehingga didapat nomor kartu

kredit yang valid dari penyelenggara yang diinginkan dan memiliki panjang nomor yang sesuai.

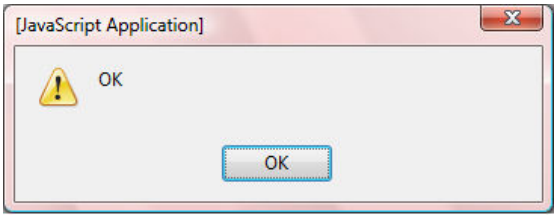
5. Fungsi `gen_VISA()`
   Fungsi ini adalah fungsi yang digunakan untuk memilih penyelenggara kartu kredit VISA.

6. Fungsi `gen_MC()`
   Fungsi ini adalah fungsi yang digunakan untuk memilih penyelenggara kartu kredit MasterCard.

3.2 cek.html



Gambar di atas adalah gambar tampilan muka dari file cek.html.



Gambar di atas akan ditampilkan bila nomor kartu kredit yang dimasukkan oleh pengguna valid. Bila tidak valid, maka pesan yang akan ditampilkan adalah "invalid".

Berikut ini akan dijelaskan mengenai kode program dari file cek.html dengan lebih rinci:

1. Fungsi checkCC()
   Fungsi ini adalah fungsi yang digunakan untuk melakukan validasi nomor kartu kredit dengan menggunakan algoritma Luhn. Parameternya berupa string yang dimasukkan oleh pengguna.

   Mula-mula, masukan pengguna diubah urutannya menjadi dari digit pali kanan ke digit paling kiri. Hal ini dilakukan untuk mengantisipasi perbedaan penerapan langkah-langkah pada subbab 2.4 pada nomor kartu kredit yang panjangnya ganjil dan genap. Selanjutnya, hal yang perlu dilakukan adalah mengubah langkah-langkah pada subbab 2.4 ke dalam operasi matematika.

   Fungsi ini akan mengembalikan status pengujian yang nilainya true atau false sehingga hasilnya dapat ditampilkan kepada pengguna.

2. Fungsi `validateForm()`
   Fungsi ini adalah fungsi yang digunakan untuk menampilkan pesan kepada pengguna mengenai validitas dari nomor kartu kredit yang dimasukkan.

3.3 Pengujian Aplikasi

Dengan menggunakan pembangkit nomor kartu kredit pada file gen.html, penulis telah membangkitkan 10 buah nomor kartu kredit dari penyelenggara kartu kredit VISA dan 10 buah nomor kartu kredit dari penyelenggara kartu kredit MasterCard. 20 buah nomor kartu kredit tersebut telah diuji dengan menggunakan algoritma Luhn yang terdapat pada file cek.html. Berikut ini adalah tabel yang berisi 20 buah nomor kartu kredit tersebut beserta status pengujiannya.

No.	Nomor Kartu	Status
1	4485992318857608 (VISA)	Lulus
2	4539021130629585 (VISA)	Lulus
3	4532994565556084 (VISA)	Lulus
4	4556928310329075 (VISA)	Lulus
5	4556214252337941 (VISA)	Lulus
6	4175615247266691 (VISA)	Lulus
7	4539407144663639 (VISA)	Lulus
8	4716357295212469 (VISA)	Lulus
9	4929484161546839 (VISA)	Lulus
10	4485424774108193 (VISA)	Lulus
11	536041826513373 (MasterCard)	Lulus
12	543883630250903 (MasterCard)	Lulus
13	518927047051569 (MasterCard)	Lulus
14	518962626126904 (MasterCard)	Lulus
15	533791281059451 (MasterCard)	Lulus
16	531895278435038 (MasterCard)	Lulus
17	531913115233621 (MasterCard)	Lulus
18	537711397961362 (MasterCard)	Lulus
19	524584616868352 (MasterCard)	Lulus
20	535875408517208 (MasterCard)	Lulus

4. KESIMPULAN

Kesimpulan dari pembahasan yang telah dilakukan tersebut adalah:

1. Dengan menggunakan pembalikan dari algoritma Luhn, dapat dibangkitkan nomor kartu kredit yang baik, sehingga kesalahan pengetikan saat memasukkan informasi dapat segera diketahui.

2. Semua nomor kartu kredit yang dibangkitkan lulus dari pengecekan dengan menggunakan algoritma Luhn

3. Algoritma Luhn dapat digunakan untuk memeriksa kesalahan saat memasukkan data nomor kartu kredit.

4. Algoritma Luhn tidak dapat memeriksa kesalahan yang terjadi lebih dari satu digit angka.

DAFTAR REFERENSI

[1] R. Munir, *Diktat Kuliah IF5054 Kriptografi*, Program Studi Teknik Informatika Institut Teknologi Bandung, 2006.

[2] http://en.wikipedia.org/wiki/Luhn_algorithm

[3] http://www.merriampark.com/anatomycc.htm