Nama : Moko Dwi Saputro

NIM : 1714321033

Kelas : Sore (E)

Mata Kuliah : Telematika

Program Study: Teknik Informatika

CRC (Cyclic Redundancy Code)

1. Pengertian CRC (Cyclic Redundancy Code)

   CRC merupakan teknik untuk mendeteksi kesalahan-kesalahan di dalam data digital, tetapi tidak bisa memperbaiki kesalahan apabila terdeteksi. Teknik ini digunakan di dalam proses pengiriman data. Pada saat pengiriman data, terkadang data yang diterima oleh penerima tidak sesuai dengan data yang dikirimkan. Kondisi ini terjadi karena adanya kerusakan data pada saat proses pengiriman data, disebabkan adanya gangguan yang biasanya terjadi pada media komunikasi. Dengan adanya kerusakan data yang mungkin terjadi, maka untuk mendeteksi kerusakan data tersebut digunakan suatu cara untuk menghitung suatu nilai terhadap data tersebut. Salah satu cara yang digunakan adalah CRC (Cyclic Redundancy Check) yang mempunyai beberapa varian bergantung pada bilangan polynomial yang digunakan dalam proses komputasinya.

2. Prinsip Kerja CRC (Cyclic Redundancy Code)

   ` Pada intinya prinsip kerja CRC adalah, kita menganggap suatu file yang kita proses sebagai suatu string yang besar, yang terdiri dari bit-bit. Dan kita operasikan suatu bilangan polynomial yang sangat besar. Untuk menghitung nilai CRC, kita membagi bilangan polynomial sebagai representasi dari file, dengan suatu bilangan polynomial kecil yang sudah terdefinisi untuk jenis varian CRC tertentu.

   Nilai CRC adalah sisa hasil bagi tersebut yang biasa disebut dengan checksum. Setiap pembagian pasti menghasilkan suatu sisa hasil bagi (meskipun bernilai 0), tetapi ada perbedaan dalam melakukan pembagian pada perhitungan CRC. Secara umum (prinsip aljabar biasa), pembagian dapat kita lakukan dengan mengurangi suatu bilangan dengan pembaginya secara terus-menerus sampai menghasilkan suatu sisa hasil bagi (yag lebih kecil dari bilangan pembagi). Dari nilai hasil bagi, sisa hasil bagi dan bilangan pembagi kita bias mendapat bilangan yang dibagi dengan mengalikan bilangan pembagi dengan hasil bagi dan menambahkan dengan sisa hasil bagi. Dalam perhitungan CRC, operasi pengurangan dan penjumlahan dilakukan dengan mengabaikan setiap carry yang didapat. Tentu saja hal ini juga akan berpengaruh pada proses pembagian yang menjadi dasar utama dalam melakukan perhitungan CRC. Operasi dalam CRC juga hanya melibatkan nilai 0 dan 1, karena secara umum kita beroperasi dalam level bit. Contoh perhitungan dalam CRC adalah sebagai berikut :

   (1) 1101 (2) 1010 1010

   1010- 1111+ 1111- ---- ---- ----

   0011 0101 0101

   Pada contoh di atas tersebut, operasi pertama (1) adalah operasi umum yang digunakan dalam operasi aljabar, yaitu dengan menghitung nilai carry yang dihasilkan. Sedangkan operasi kedua (2) adalah operasi dasar yang akan digunakan dalam proses perhitungan nilai CRC. Nilai carry diabaikan, sehingga operasi pengurangan dan penjumlahan akan menghasilkan suatu nilai yang sama. Kedua operasi ini bisa kita lakukan dengan operasi XOR. Pada proses pembagian yang dilakukan akan tampak sekali bedanya, karena pengurangan yang dilakukan sama seperti malakukan penjumlahan. Nilai hasil bagi diabaikan, jadi hanya sisa hasil bagi (remainder) yang diperhatikan. Dan remainder inilah yang akan menjadi dasar bagi

nilai CRC yang dihasilkan.

3. Perhitungan CRC (Cyclic Redundancy Code) Secara Aljabar

Untuk melakukan perhitungan CRC terhadap suatu data, maka yang pertama diperlukan adalah suatu bilangan polinom yang akan menjadi pembagi dari data yang akan diolah (disebut sebagai poly). Dan kita juga menghitung lebar suatu poly, yang merupakan posisi biat tertinggi. Misalkan kita sebut lebar poly adalah W, maka jika kita mempunyai poly 1001, makam W poly tersebut adalah 3. Bit tertinggi ini harus kita pastikan bernilai 1. Dengan mengetahui W secara tepat sangat penting, karena berpengaruh pada jenis CRC yang akan digunakan (CRC 16, CRC 32, dan lain-lain). Data yang diolah mungkin saja hanya beberapa bit saja, lebih kecil dari nilai poly yang akan digunakan. Hal ini akan menyebabkan kita tidak mengolah semua nilai poly yang telah ditentukan. Untuk mengatasi hal tersebut, dalam perhitungan dasar secara aljabar, kita menambahkan suatu string bit sepanjang W pada data yang akan kita olah. Tujuannya untuk menjamin keseluruhan data dapat kita olah dengan benar.

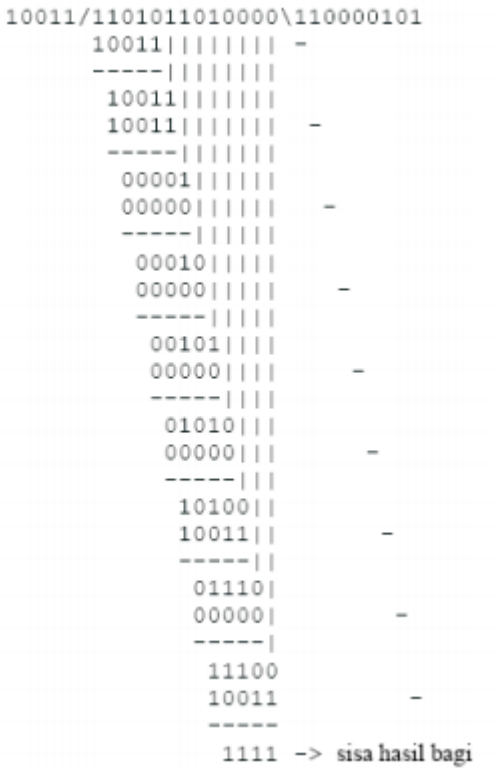
Contoh perhitungan sebagai berikut :

Poly = 10011

(Width W = 4)

Bitstring + W zeros = 110101101 + 0000

Contoh pembagian yang dilakukan :



Nilai remainder inilah yang menjadi nilai CRC. Pada proses pembagian di atas, kita mendapat hal penting yang perlu diperhatikan dalam perhitungan secara aljabar ini adalah kita tidak perlu melakukan operasi XOR ketika bit tertinggi bernilai 0, tapi kita hanya melakukan pergeseran \(shift) sampai didapat bit tertinggi yang bernilai 1. Hal ini akan sedikit mempermudah dan mempercepat operasi aljabar. Secara notasi bias dituliskan sebagai berikut :

$a(x).xN = b(x).p(x) + r(x)$

Keterangan :

a(x) :Bilangan polynomial yang

merepresentasikan data.

xN : Nilai 0 sebanyak W

b(x) : hasil bagi yang didapat

p(x) : poly

r(x) : sisa hasil bagi, nilai CRC

Karena nilai CRC adalah sisa hasil bagi, maka untuk mengecek integritas data dapat dilakuka dengan beberapa cara, diantaranya :

a. Kita hitung nilai CRC dari data yang asli,

lalu kita cocokkan dengan nilai CRC yang disimpan (di append dengan data). Datayang asli mudah kita dapatkan karena nilai CRC sepanjang N-1.

b. Data yang asli kita tambah dengan nilai CRC, lalu kita bagi dengan nilai poly, maka sisa hasil bagi adalah 0 jika data benar.

4. Pendekatan Tabel CRC (Cyclic Redundancy Code)

Perhitungan nilai CRC yang berbasis bit akan sangat lama dan tidak efisien. Cara tersebut dapat diperbaiki dengan melakukan operasi dengan basis byte. Poly yang digunakan akan dioperasikan dalam bentuk byte, sehingga harus panjang kelipatan 8 bit (byte). Misalkan sebagai contoh dalam memproses CRC 8 bit (1 byte) agar lebih sederhana. Register akan berisi 8 bit, dan pada sekali shift akan menggeser sebanyak 4 bit.

Misalkan isi awal register : 10110100 Kemudian 4 bit pertama akan digeser (1011), dan memasukkan 4 bit baru misalkan (1101). Maka 8 bit register sekarang adalah :

01001101 4 bit yang digeser (top bits) : 1011

Polynomial yang digunakan (W=8) : 101011100

Langkah selanjutnya adalah dengan melakukan XOR terhadap isi register dengan polynomial. Top Register

1011 01001101

1010 11100 (*1) operasi XOR dimulai pada bit tertinggi

----------+ bernilai 1.

0001 10101101 hasil XOR

Karena Top (isi register yang digeser keluar) masih

mengandung nilai 1, maka ulangi lagi langkah di

atas.