Enkripsi Reed – Solomon Code untuk Toleransi Kesalahan Data

Satria Priambada - 13513034
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
satria.priambada@students.itb.ac.id

Abstract*—Dalam dunia digital saat ini begitu banyak proses pertukaran dan penyimpanan data yang dilakukan. Dalam proses ini umumnya dapat terjadi gangguan atau kerusakan. Bila terjadi kerusakan data maka akan terjadi kesalahan penyampaian pesan. Makalah ini akan menjelaskan mengenai bagaimana proses toleransi kesalahan dapat dilakukan dengan menggunakan Reed-Solomon *Coding

Keywords*— Reed-Solomon Code, Toleransi kesalahan, Enkripsi Data, RAID , Galois Field, *Cyclic Code

I. PENDAHULUAN

Saat ini begitu banyak hal berkaitan dengan dunia digital. Hal ini menyebabkan begitu banyak data harus disimpan dan diatur. Data dapat disimpan dengan berbagai cara. Salah satu metode yang saat ini digunakan adalah sistem RAID[6] *(Redundant Arrays of Inexpensive Disks).* Sistem ini melakukan penulisan data secara berulang pada disk dengan tujuan untuk memastikan bahwa data yang akan diambil merupakan data yang benar. Bila terjadi kesalahan proses penulisan berulang akan memungkinkan terjadinya proses pemulihan *(recovery)* data.

Dalam sistem ini dilakukan penggabungan terhadap banyak disk dengan ukuran kecil yang memiliki harga murah namun memiliki kemampuan penyimpanan data yang cukup tinggi. Sebagian besar sistem jaringan saat ini dibuat berdasarkan cara kerja sistem ini. Namun dengan bertambahnya disk yang digunakan maka kemungkinan terjadinya kerusakan pada data semakin besar. Bila terdapat satu bagian disk yang rusak maka arti dan makna dari keseluruhan data dapat menjadi rusak / berubah. Oleh sebab itu perlu adanya kemungkinan toleransi kesalahan .

Salah satu cara yang dapat digunakan untuk melakukan toleransi kesalahana adalah *"n +1-parity."* [6] / *Check bits*[1]. Cara ini melakukan penambahan bit sebagai pengecek pada akhir data. Nilai bit yang ditulis merupakan hasil dari XOR *(exclusive - or)* dari bit ke n dari setiap tempat penyimpanan data. Bila terdapat data yang rusak, maka dapat dilakukan rekonstruksi data tersebut dari membalik tersebut dari membalik proses XOR dari bit ke n tempat penyimpanan data yang lainnya.

Cara ini mudah untuk diaplikasikan, namun cara ini membutuhkan satu buah tempat penyimpanan data tambahan yang beisi bit pengecek kesalahan. Kekurangan utama dari cara ini adalah proses rekonstruksi data hanya dapat dilakukan jika hanya 1 tempat penyimpanan yang rusak. Bila terjadi beberapa kesalahan pada beberapa tempat penyimpanan maka tidak dapat dilakukan proses pemulihan data.

Dengan berlatar belakang permasalahan ini, pada tahun 1960, 2 orang ilmuawan bernama Irving S. Reed dan Gustave Solomon memperkenalkan algoritma baru yang disebut Reed-Solomon Code[4]. Cara yang mereka kembangakn merupakan sub - bagian dari cara yang lain yaitu BCH Codes (Bose, Chaudhurj, dan Hocquenghem), pada prakteknya Reed – Solomon code lebih sederhana dan mudah dipelajari bagi penulis.

Dalam kehidupan sehari – hari Reed-Solomon Code digunakan dalam berbagai metode penyimpanan data dalam jaringan komputer dan berbagai jenis CD. Penyampaian komunikasi wireless dan *HDTV (High Definition TV)* juga mengunakan Reed-Solomon Code [4],[5]. Aplikasi lainnya adalah sebagai toleransi kesalahan dari *barcode* dan QR Code[3]. Pada QR Code toleransi kesalahan yang dapat dilakukan tergantung dari proses pengkodean yang dilakukan, dimana pada tingkat kode paling sederhana QR Code dapat menerima persentase kesalahan hingga 5% dan pada tingkat kode yang semakin dimampatkan tingkat koreksi kesalahan yang dapat dilakukan dapat mencapai 30%.

Selain itu algoritma ini juga membantu proses komunikasi satelit *Voyager II* dari luar angkasa ke bumi[4]. Para ilmuan dari *Jet Propulsion Laboratory(JPL)* mencoba menggunakan Reed-Solomon Code (1986) dan berhasil mendapatkan 21.600 bits setiap detik dari sinyal yang berjarak sangat jauh (Uranus ke Bumi kurang lebih 3.2 miliar kilometer pada aphelion (jarak terjauh dengan matahari))dengan energi sinyal yang 100 miliar kali lebih lemah dari baterai jam tangan.

II. TEORI BILANGAN

A. Teori Sifat Pembagian (Prima)

Terdapat 2 buah sifat pembagian yaitu [1]:

(1) Habis membagi (a | b) dibaca a habis membagi b. Pernyataan tersebut dapat ditulis dalam bentuk persamaan :

$$b = c \cdot a$$

b hasil kali dari a dan c, saat a dan c bilangan bulat

(2) Tidak habis membagi bila syarat 1 tidak terpenuhi

Dalam matematika bilangan yang tidak memiliki bilangan yang dapat habis membagi bilangan tersebut kecuali 1 dan bilangan itu sendiri disebut bilangan prima.

B. Aritmatika Modular

Aritmatika modular adalah salah satu bagian dari teori bilangan yang hanya mempedulikan sisa dari hasil pembagian sebuah bilangan [2]. Praktek dalam kehidupan sehari – hari misal saat sedang menghitung jam pada dasarnya sedang dilakukan operasi X mod 24 dimana X adalah jam yang diinginkan. Contoh : 100 jam ke depan bila saat ini pk. 10.00 adalah 100 mod 24 = 4. Maka didapatkan bahwa jawaban persoalan diatas adalah pk. 14.00. Operasi ini sangat membantu dalam kasus penghitungan yang berulang.

Dalam perhitungan nilai ini terdapat sebuah teorema yang disebut Teorema Eucledian. Teorema ini digunakan untuk mencari nilai FPB (faktor pembagi terbesar). Misalkan *m* dan *n* adalah bilangan bulat dengan nilai *n* lebih besar dari 0, maka jika *m* dibagi *n* akan terbentuk q(nilai pembagian / *quotient*) dan r (sisa pembagian/ *remainder*). Pernyataaan dapat ditulis :

$$m = n \cdot q + r$$

Bila pembagian memenuhi persamaan diatas terdapat teorema yang menyatakan bahwa FPB(m,n) = FPB(n,r). Dalam menentukan FPB bilangan *m* dan *n* maka akan dilakukan pembagian berturut – turut untuk mencari nilai sisa pembagian 0 <= r < n Contoh :

Input = m >= n > 0 Output = r_n

$R_0 = m$

$R_1 = n$

$R_0 = q_1 \cdot R_1 + R_2$ syarat($0 < R_2 =< R_1$)

$R_1 = q_2 \cdot R_2 + R_3$ syarat($0 < R_3 =< R_2$)

(...)

$R_{n-2} = q_{n-1} \cdot R_{n-1} + R_2$ syarat($0 < R_n =< R_{n-1}$)

$R_{n-1} = q_n \cdot R_n + 0$ hingga r mencapai 0

Menurut teorema FPB(m,n) = FPB(r_o,r_1) = (...) = FPB(r_n,0) = r_n. Maka FPB terbesar adalah r_n. Bilangan yang didaptakan sebagai hasil adalah bilangan prima. Dari persamaan diatas dapat diamati bahwa nilai *r* adalah hasil dari persamaan *m* mod *n.*

C. Kongruensi

Pada akhir abad ke - 18 Karl Friedrich Gauss mengemukakan konsep baru yang disebut kongruensi.[2] Sifat ini menyatakan bahwa dua bilangan memiliki hasil mod yang sama. Contoh 11 mod 12 kongruen dengan 35 mod 12 dst.

Dari sifat yang telah ada maka dapat dibuat sebuah penurunan jika *m* bilangan bulat positif. Jika $a \equiv b \pmod m$ dan $c \equiv d \pmod m$ maka berlaku

$$a + c \equiv b + d \pmod m$$

$$ac \equiv bd \pmod m.$$

Pembuktian dapat dilihat pada [2].

D. Modulo Inverse

Modulo inverse adalah sifat yang berlaku untuk operasi modular saat *a* dan *m* relatif prima dengan nilai *m* > 1 maka inverse dari *a* (mod *m*) adalah *x*

$$x.a \equiv 1 \pmod m.$$

atau

$$a^{-1} \pmod m \equiv x$$

Contoh : $4^{-1} \pmod 9$ maka

9.1 + x . 4 = 1

nilai x = -2,7,dst. Karena sifat kongruen tetap berlaku.[2]

III. ENKRIPSI REED-SOLOMON CODE

A. Struktur Pesan

Sebuah Reed-Solomon Code ditulis dalam notasi (n,k,d) dan akan memiliki bentuk sebagai kumpulan data sebagai berikut :

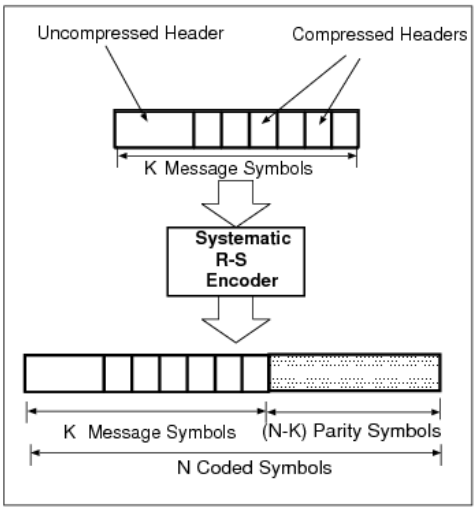


Figure 2. Reed-Solomon encoding of packet headers.

From : http://www.utdallas.edu/~aria/mcl/headercomp/

Bila terdapat sejumlah pesan dengan ukuran k maka akan dikirimkan pesan sepanjang n yang berisis pesan sepanjang k dan simbol parity sepanjang n – k. Namun jumlah simbol parity ini juga dapat dinyatakan dalam pernyataan d = 2*t*+1 [5] dengan *t* merepresentasikan jumlah error maksimal yang dapat dikoreksi dan d menyatakan jumlah simbol paling sedikit untuk dapat mengatasi error sebanyak *t*. Untuk mengatasi 2 error sebuah kode membutuhkan *d =(2).2 +1* = 5. Nilai d disebut sebagai *parity check* dari sebuah kode. Semakin panjang d maka semakin banyak error yang dapat

diatasi, namun semakin sedikit informasi yang terdapat dalam 1 buah pesan.

Panjang dari pesan $n = 2^m-1$ dimana nilai m adalah jumlah bit yang akan merepresentasikan setiap simbol. Misalkan akan dikirim pesan dengan panjang $n = 255$ simbol, maka setiap simbol akan direpresentasikan dengan (2^8) bit, $m = 8$.

B.Galois Field

Galois Field adalah sebuah *Field* terbatas yang akan dijadikan tempat untuk mengkodekan seluruh kode biner yang diinginkan. *Field* sendiri berarti sekumupulan elemen yang dapat dioperasikan berdasarkan operasi dasar yaitu penjumlahan, pengurangan, perkalian. Dalam kasus Galois Field penjumlahan dilakukan dengan melakukan operasi dari XOR *(exclusive - or).* Hal ini menyebebakan hasil dari penjumlahan dan penurangan sama. Terbatas berarti *Field* memiliki batas ukuran tertentu yang didefinisikan terlebih dahulu[4].

Notasi dari Galois Field ditulis dalam bentuk $GF(p^m)$. Nilai p harus merupakan sebuah bilangan prima dan nilai $m >= 0$. Pada kasus biner nilai $p = 2$ dan m adalah banyaknya bit yang merepresentasikan simbol. Dalam pembentukan Galois Field diperlukan sebuah primitif polinomial yang berderajat sesuai yang dibutuhkan. Primitif polinomial adalah sebuah polinomial yang memilki sifat prima, yaitu tidak dapat dibagi oleh polinomial lain. Ide dari field berisi polinomial ini adalah melakukan peletakan polinomial sebagai representasi kode biner dengan kemampuan yang dapat berulang. Misal saat ingin merepresentasikan 1001 maka dengan simbol polinomial dapat ditulis $x^3 + 1$. Agar lebih jelas perhatikan contoh $m = 8$ pada tabel dibawah ini :

TABEL KONVERSI BITS KE SIMBOL GALOIS

Generated Element	Polynomial Element	Binary Element	Decimal Element
0	0	0000	0
x^0	1	0001	1
x^1	x	0010	2
x^2	x^2	0100	4
x^3	x^3	1000	8
x^4	$x+1$	0011	3
x^5	x^2+x	0110	6
x^6	x^3+x^2	1100	12
x^7	x^3+x+1	1011	11
x^8	x^2+1	0101	5
x^9	x^3+x	1010	10
x^{10}	x^2+x+1	0111	7
x^{11}	x^3+x^2+x	1110	14
x^{12}	x^3+x^2+x+1	1111	15
x^{13}	x^3+x^2+1	1101	13
x^{14}	x^3+1	1001	9
x^{15}	1	0001	1

From : http://web.eecs.utk.edu/~**plank**/**plank**/papers/CS-96-332.pdf

Dapat diamati dari tabel bahwa penambahan 1 buah elemen menggeser elemen sebelumnya sebanyak satu buah ke kanan (Bandingkan *binary element* saat indeks *generated element* x^0 dengan x^1). Bila penggeseran habis maka akan diulangi lagi dari awal dengan menambahkan satu bit pada kolom sebelahnya (Bandingkan *binary element* saat indeks *generated element* x^0 dengan x^4). Saat sudah mencapai x^{15} maka nilai dari *decimal element* akan kembali menjadi 1. Hal ini menyebabkan Reed-Solomon Code disebut sebagai *Cyclic Code*[4]. Pada kolom *polynomial element* dapat diamati bahwa setiap *generated element* akan selalu direpresentasikan dalam bentuk primitif polinomial.

Keberadaan nilai *generated element* yang akan selalu direpresentasikan dalam bentuk primitif polinomial dapat dilakukan dengan operasi modulo pada fungsi dengan urutan yang diinginkan dengan primitif elemen q(x)[7]. Nilai q(x) akan berbeda pada setiap pendefinisian Galois Filed. Pada $m = 3$ maka nilai q(x)

$$q(x) = x^3 + x + 1$$

Nilai ini berlaku sebagai pembagian modulo yang menjadi dasar pengulangan dari kode. Berikut adalah tabel sederhana hingga $m = 32$[7].

TABEL POLINOMIAL BASIS MODULO

m = 4	$q(x) = x^4 + x + 1$
m = 8	$q(x) = x^8 + x^4 + x^3 + x^2 + 1$
m = 16	$q(x) = x^{16} + x^{12} + x^3 + x + 1$
m = 32	$q(x) = x^{32} + x^{22} + x^2 + x + 1$

From : http://web.eecs.utk.edu/~**plank**/**plank**/papers/CS-96-332.pdf

Besarnya nilai kolom *polynomial element* (pEl(x)) akan berasal dari *generated element* gEl(x) dengan persamaan sebagai berikut :

$$pEl(x) = gEl(x) \bmod q(x)$$

Nilai elemen *x* yang didaptakan dari sini akan menjadi koefisien dari persamaan berikutnya dan selanjutnya akan disebut sebagi α.

C.Generator Pesan

Dalam pembuatan pesan Reed-Solomon Code diperlukan sebuah fungsi yang merupakan generator dari pesan yaitu g(x). Fungsi ini memiliki definisi [5]:

$$g(x) = (X - \alpha^1)(X - \alpha^2)(X - \alpha^3) \ldots (X - \alpha^{2t})$$

Saat proses enkripsi dilakukan maka pesan akan dibagi dengan g(x) untuk membentuk *parity* pesan. Generator pesan ini memiliki fungsi untuk mengecek apakah sebuah pesan yang diterima sudah merupakan pesan yang benar atau tidak.

Bila nilai x disulihkan (substitusi) dengan α maka hasil pengurangan (XOR) dari 2 buah bilangan yang sama akan menghasilkan nilai 0 pada semua tempat. Namun bila x diganti dengan α dan tidak semua hasil menunjukkan angka 0 berarti terdapat kesalahan dalam pesan yang masih perlu diperbaiki. Generator berfungsi

pada saat melakukan enkripsi dan dekripsi. Sebuah Reed-Solomon Code dengan dimensi tertentu akan memilki generator sendiri. Panjangnya polinomial generator sangat bergantung pada tingkat kesalahan (*error*) yang ingin ditangani. Semakin besar tingat kesalahan yang ingin ditangani maka fungsi g(x) akan semakin panjang.

D.Proses Enkripsi

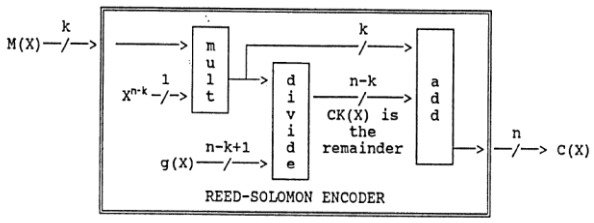
Data yang akan dikirim / disimpan akan dienkripsi. Ambil contoh pesan telah dijadikan polinom *m(x)*. Generator pesan adalah fungsi g(x), Hasil pembagian (*quotient*) adalah fungsi q(x), sisanya (*parity*) adalah p(x) , dan kode yang diinginkan C(x). Enkripsi akan mengikuti aturan sebagai berikut [4]:

1. Kalikan m(X) dengan X^{n-k}, X adalah elemen m(x) pada urutan tertentu yang ingin dienkripsi
2. Bagi hasil perkalian tersebut untuk mendapatkan sisa (*parity*) dari pesan.
3. Bentuk pesan dengan menjumlahakan sisa (*parity*) dengan hasil kali m(X) . X^{n-k}

Dalam persamaan matematika dapat ditulis :

1. $m(X) . X^{n-k} = q(X). g(X) + p(X)$
2. $p(X) = m(X) . X^{n-k} \mod g(X)$
3. $C(X) = m(X) . X^{n-k} + p(X)$

Kegunaan perkalian dengan A adalah untuk menggeser pesan 1 per 1 ke kanan sehingga pesan tidak akan saling menimpa. Dalam perkalian dan penjumlahan dalam Galois Field menggunakan XOR, perkalian $a_i . a_j = a_{i+j}$. Berikut adalah gambar algoritma enkripsi :



from : http://jeffareid.net/misc/msc-21834.pdf

Cotoh :

Jika ingin mengirim pesan sepanjang *n* = 7 dengan jumlah informasi *k* = 3, maka

$7 = (2^m-1)$; m = 3 bit representasi

Code rate = k/n = 3/7 x100% = 42.85%

$n - k = 4$

Error Correction Capability =

$(n - k) / 2n$ = 4/7 x100% = 67.15%

Primitif polinomial : $1+x+x^3$

Generator Pesan :

$g(x) = [(X - \alpha) (X - \alpha^2)][(X - \alpha^3) (X - \alpha^4)]$

$g(x) = (X^2 - (\alpha+ \alpha^2)X + \alpha^3) (X^2 - (\alpha^3+ \alpha^4)X + \alpha^7)$

$g(x) = (X^2 - (\alpha^4)X + \alpha^3) (X^2 - (\alpha^6)X + \alpha^0)$ {lihat tabel}

$g(x) = X^4 - (\alpha^6+ \alpha^4)X^3 + (\alpha^0 + \alpha^3 + \alpha^{10}) X^2 - (\alpha^2+ \alpha^4)X + \alpha^3$ {plus minus sama (XOR), perhitungan liat tabel}

$g(x) = X^4 - \alpha^3X^3 + X^2 - \alpha X + \alpha^3$

TABEL KONVERSI BITS KE SIMBOL GALOIS

no	Bit 1	Bit 2	Bit 3	Desimal
0	0	0	0	0
1	0	0	1	1
α	0	1	0	2
α^2	1	0	0	4
α^3	0	1	1	3
α^4	1	1	0	6
α^5	1	1	1	7
α^6	1	0	1	5

Misalkan pesan : 000(0) 111(α_5) 000(0).

Bentuk yang akan dijadikan kode :

$m(X) = 0 X^2 + \alpha^5 X + 0$

$C(X) = m(X) . X^{n-k} + p(X)$

$C(X) = \alpha^5 X^5 + \mathbf{(\alpha^5X^5 \pmod{g(x)})}$

Lakukan mod polinomial pada fungsi sebagai berikut :

$X^4 - \alpha^3X^3 + X^2 - \alpha X + \alpha^3 \,/\, \overline{\alpha^5X^5}$

$\alpha^5X^5 - \alpha X^4 + \alpha^5X^3 - \alpha^6X^2 + \alpha X$

$\alpha X^4 - \alpha^5X^3 + \alpha^6X^2 - \alpha X$

$\alpha X^4 - \alpha^4X^3 + \alpha X^2 - \alpha^2X + \alpha^4$

Hasil bagi = $\alpha^5X + \alpha$

Sisa (mod) = $(\alpha^4 + \alpha^5) X^3 + (\alpha+\alpha^6) X^2 + (\alpha^2+ \alpha) X + \alpha^4$ {plus dan minus sama merupakan XOR}

Sisa (mod) = $X^3 + \alpha^5 X^2 + \alpha^4 X + \alpha^4$ {lihat tabel}

$$C(X) = \alpha^5 X^5 + X^3 + \alpha^5 X^2 + \alpha^4 X + \alpha^4$$

Bila ditulis dalam bentuk biner maka:

$$C(X) = 000\ 111\ 000\ 001\ 111\ 110\ 110$$

Untuk melihat contoh yang lebih kompleks (GF(16)) dapat ditemukan pada [4].

IV. DEKRIPSI REED-SOLOMON CODE

A. Konsep

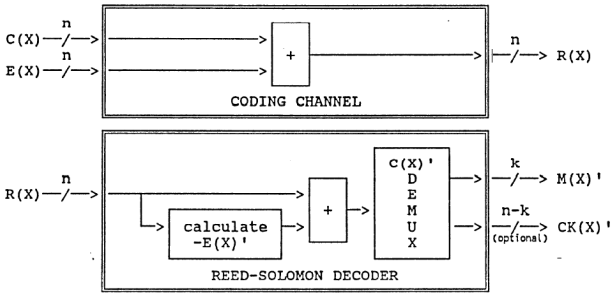
Dalam melakukan dekripsi Reed-Solomon Code hal yang pertama ingin dilakukan adalah mengecek *parity* dari pesan yang diterima. Bila ternyata pesan yang diterima memiliki kesalahan maka hal yang harus dilakukan adalah mencari sindrom dari matrix atau S(X). Dari S(X) akan dicari kumpulan error yang disebut dengan E(X). Dengan melakukan proses pengurangan (XOR) kesalahan (E(X)) pada pesan yang diterima (R(X)) maka pesan C(X) akan ditemukan kembali. Selama jumlah kesalahaan tidak melebihi toleransi (t) yang diberikan maka pesan akan selalu dapat dikembalikan dari R(X) menjadi C(X). Proses ini pada umumnya merupakan proses yang jauh lebih rumit dibandingkan membuat enkripsi.

B. Langkah Dekripsi

Terapat 5 langkah dekripsi [4] :

1. Hitung lokasi dari syndrom
2. Tentukan kata pembuat kesalahan dalam syndrom
3. Tentukan lokasi kesalahan dari kata pembuat kesalahan
4. Hitung nilai dari E(X) dari lokasi kesalahan
5. Hitung kode sebenarnya dengan menggunakan pengurangan R(X) – E(X) = C(X).

Proses pencarian ini harus dilakukan berurutan. Pada dasarnya proses ini melakukan pencarian dari kumpulan terbesar dan kemudian semakin spesifik dengan berjalannya proses. Hal ini merupakan salah satu keunggulan Reed-Solomon Code yaitu kode dapat diketahui bit pembuat kesalahan dan perbaikan dapat dilakukan hingga komponen terkecil data dengan banyak kesalahan boleh berturut – turut sebanyak *t*.



from : http://jeffareid.net/misc/msc-21834.pdf

Cara penentuan sindrom ditentukan dengan 2 cara yaitu [4] :

$$S[i] = R[ai] \ (=E(aI))$$

atau

$$S(X) = R(X) \bmod g(X) = \text{sisa } [R(X)/g(X)]$$

Nilai *i* akan terus naik hingga mencapai *k* yang berarti bahwa setiap bit dalam kode akan dicek dan bila terdapat nilai yang bukan 0 maka posis akan dicatat. Selanjutnya proses yang sama dilakukan didalam himpunan sindrom untuk mencari tempat kesalahan. Proses akan dilakukan sekali lagi untuk mencari bit yang salah. Setelah bit ditemukan akan dilakukan proses penghitungan untuk menentukan besarnya kesalahan yang akan dikurangi dari R(X).

Penentuan posisi sindrom yang akan dilakukan dalam proses ini dapat dikerjakan dengan 2 buah metode yaitu *Chien Search Method* dan *Explicit Method* [4]. Sementara untuk penentuan tempat kesalahan akan digunakan *Berlekamp's Algorithm* [4]. Metode – metode tersebut akan menggunakan matriks.

Contoh :

C(X) = 000 111 000 001 111 110 110

R(X) = 000 **101** 000 001 111 110 110

$n = 7$ dan $k = 3$

$n - k = 4$

$g(x) = X^4 - \alpha^3 X^3 + X^2 - \alpha X + \alpha^3$

$R(X) = \alpha^6 X^5 + X^3 + \alpha^5 X^2 + \alpha^4 X + \alpha^4$

Sulihkan X dengan α. Lakukan proses ini hingga $n - k$ (4) kali dimulai dari 1 agar seluruh elemen pesan melewati pengecekan.

$$S_1 = R(\alpha) = \alpha^6 \alpha^5 + \alpha^3 + \alpha^5 \alpha^2 + \alpha^4 \alpha + \alpha^4$$
$$= \alpha^{11 \bmod 7 = 4} + \alpha^3 + \alpha^{7 \bmod 7 = 0} + \alpha^5 + \alpha^4 = \alpha^6$$
$$S_2 = R(\alpha^2) = \alpha^6 \alpha^{2.5} + \alpha^{2.3} + \alpha^5 \alpha^{2.2} + \alpha^4 \alpha^2 + \alpha^4$$
$$= \alpha^{16 \bmod 7 = 2} + \alpha^6 + \alpha^{15 \bmod 7 = 6} + \alpha^6 + \alpha^4 = \alpha^6$$
$$S_3 = R(\alpha^3) = \alpha^6 \alpha^{3.5} + \alpha^{3.3} + \alpha^5 \alpha^{2.3} + \alpha^4 \alpha^3 + \alpha^4$$
$$= \alpha^{21 \bmod 7 = 0} + \alpha^{9 \bmod 7 = 2} + \alpha^{11 \bmod 7 = 4} + \alpha^{7 \bmod 7 = 0} + \alpha^4 = \alpha^2$$
$$S_4 = R(\alpha^4) = \alpha^6 \alpha^{4.5} + \alpha^{4.3} + \alpha^5 \alpha^{2.4} + \alpha^4 \alpha^4 + \alpha^4$$
$$= \alpha^{26 \bmod 7 = 5} + \alpha^{12 \bmod 7 = 5} + \alpha^{13 \bmod 7 = 6} + \alpha^{8 \bmod 7 = 1} + \alpha^4 = \alpha^2$$

Selanjutnya dilakukan pencarian terhadap tempat kesalahan dengan menggunakan matrixs dengan L1 dan L2 sebagai lokasi yang ingin dicari :

$$\begin{matrix} S1 & S2 \\ S2 & S3 \end{matrix} \times \begin{matrix} L1 \\ L2 \end{matrix} = \begin{matrix} S3 \\ S4 \end{matrix}$$

Lakukan inverse terhadap matriks 2x2 untuk mendapatkan nilai L1 dan L2. Lakukan penyulihan hingga membentuk matriks sebagai berikut :

$$\begin{matrix} L1 \\ L2 \end{matrix} = \begin{matrix} \alpha^3 & \alpha^0 \\ \alpha^0 & \alpha^0 \end{matrix} \times \begin{matrix} \alpha^2 \\ \alpha^2 \end{matrix}$$

$$L1 = \alpha^5 + \alpha^2 = \alpha^6 ; \ L2 = = \alpha^2 + \alpha^2 = 0$$
$$L(X) = \alpha^0 + L1.X + L2.X^2$$
$$L(X) = \alpha^0 + \alpha^6 X + 0 X^2$$
$$L(X) = \alpha^0 + \alpha^6 X$$

Lakukan proses penyulihan sekali lagi pada L(X) sebanyak *n* kali dimulai dari 0:

$$L_1 = L(\alpha^0) = \alpha^0 + \alpha^6 . \alpha^0$$
$$= \alpha^0 + \alpha^6 = \alpha^2$$
$$\mathbf{L_2 = L(\alpha) = \alpha^0 + \alpha^6 . \alpha}$$
$$\mathbf{= \alpha^0 + \alpha^{7 \bmod 7 = 0} = 0}$$
$$L_3 = L(\alpha^2) = \alpha^0 + \alpha^6 . \alpha^2$$
$$= \alpha^0 + \alpha^{8 \bmod 7 = 1} = \alpha^3$$
$$L_4 = L(\alpha^3) = \alpha^0 + \alpha^6 . \alpha^3$$
$$= \alpha^0 + \alpha^{9 \bmod 7 = 2} = \alpha^6$$
$$L_5 = L(\alpha^4) = \alpha^0 + \alpha^6 . \alpha^4$$
$$= \alpha^0 + \alpha^{10 \bmod 7 = 3} = \alpha$$
$$L_6 = L(\alpha^5) = \alpha^0 + \alpha^6 . \alpha^5$$
$$= \alpha^0 + \alpha^{11 \bmod 7 = 4} = \alpha^5$$
$$L_7 = L(\alpha^6) = \alpha^0 + \alpha^6 . \alpha^6$$
$$= \alpha^0 + \alpha^{12 \bmod 7 = 5} = \alpha^4$$

Nilai $L_2 = 0$ berarti nilai $B = X^2$ adalah error. Posis L ditentukan dengan $1/B = X^{-2 \bmod 7 = 5}$. Nilai dari E(X) adalah **α (010).** Dalam bentuk fungsi E(X) ditulis sebagai

$$E(X) = \alpha X^5$$

Langkah terakhir dalam merekonstruksi pesan adalah menentukan C(X)

$$C(X) = R(X) + E(X)$$
$$C(X) = \alpha^6 X^5 + X^3 + \alpha^5 X^2 + \alpha^4 X + \alpha^4 + \alpha^1 X^2$$
$$C(X) = (\alpha + \alpha^6) X^5 + X^3 + \alpha^5 X^2 + \alpha^4 X + \alpha^4$$
$$C(X) = \alpha^5 X^5 + X^3 + \alpha^5 X^2 + \alpha^4 X + \alpha^4$$

Pesan C(X) telah berhasil dipulihkan *(recovery)* dengan kesalahan sebanyak 1. Dalam contoh kesalahan maksimal adalah 2.

Untuk melihat contoh yang lebih kompleks (GF(16)) dapat ditemukan pada [4].

V. LAMPIRAN

Untuk penjelasan mengenai Reed-Solomon Code dalam bentuk video dan penjelasan dalam kelas dapat dilihat dari tautan berikut :

1. What is Forward Error Correction (FEC) - OTN - Reed Solomon 255,239 - YouTube. From : https://www.youtube.com/watch?v=1Nlyb0rJdIY
2. Mod-01 Lec-13 BCH and RS Code I, Coding Theory by Dr. Andrew Thangaraj, Department of Electronics & Communication Engineering, IIT Madras. From : https://www.youtube.com/watch?v=3o-R0wOu2uQ.
3. Mod-01 Lec-14 BCH and RS Code II, Coding Theory by Dr. Andrew Thangaraj, Department of Electronics & Communication Engineering, IIT Madras. From : https://www.youtube.com/watch?v=mCknpO84C28
4. Reed-Solomon Codes - I Video Lecture, MIT Lecture 10 Reed-Solomon Codes I, MIT Course , Prof. David Forney . From : http://freevideolectures.com/Course/2177/Principles-of-Digital-Communication-II/10
5. Encoding Reed Solomon Matlab by Jaka Arya Pradana. From : https://www.youtube.com/watch?v=GABzpTftj04
6. Decoding Reed Solomon Matlab by Jaka Arya Pradana. From : https://www.youtube.com/watch?v=t-ilYG_BghI

VI. TERIMA KASIH

Penulis mengucapkan terima kasih kepada Tuhan YME untuk segala berkat dan penyertaan-Nya selama proses penulisan makalah. Penulis juga mengucapkan terima kasih kepada dosen pembimbing, Bapak Rinaldi Munir, untuk segala pengajaran yang diberikan mengenai Matematika Diskrit, terkhusus mengenai Teori Bilangan yang menjadi salah satu dasar penting dalam penulisan makalah ini.

REFERENSI

[1] S. S.Epp, “Discrete Mathematics with Applications” 4th ed. Canada: Richard Stratton/Cengage Learning. , 2010, pp. 170-180

[2] K. H. Rosen, “Discrete Mathematics and its Applications” 7th ed. New York: McGraw-Hill, 2007, pp. 237 - 244

[3] Denso-Wave. "QR Code features". Archived from the original on 2012-09-15. Dibaca pada 3 Des 2014.

[4] Lyndon B. Johnson Space Center. “Tutorial on Reed-Solomon Error Correction Coding”. Houston, Texas : NASA- Tech Briefs, MSC-21834. from http://jeffareid.net/misc/msc-21834.pdf. Dibaca pada 4 Des 2014.

[5] Duke University Dep. CS. “Reed-Solomon Code” .Durham : North Carolina. from : https://www.cs.duke.edu/courses/spring10/cps296.3/rs_scribe.pdf. Dibaca pada 4 Des 2014

[6] Multimedia Communication Laboratory. “Efficient Package Data Transmission with Packet Header Compression”. Dallas : University of Texas. From : http://www.utdallas.edu/~aria/mcl/headercomp/. Dibaca pada 5 Des 2014

[7] J. S. Plank. “A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like System”. Technical Report CS-96-332 Department of Computer Science : University of Tennessee. from : http://web.eecs.utk.edu/~plank/plank/papers/CS-96-332.pdf. Dibaca pada 4 Des 2014.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 5 Desember 2014

Satria Priambada - 13513034