Implementasi *Encode* dan *Decode* pada Kode BCH (Bose-Chaudhuri-Hocquenghem) Menggunakan *Python*

Ahmad Dhiya Al Fakhri[1], Putranto Hadi Utomo[2], Bowo Winarno[3]
[1,2,3]Universitas Sebelas Maret

Abstract. Penelitian kali ini akan membahas terkait implementasi *encode* dan *decode* pada kode BCH (Bose-Chaudhuri-Honcquenghem) menggunakan bahasa pemrograman *python.* Kode BCH adalah salah satu kode untuk mengoreksi eror. Sebelum diimplementasikan kepada bahasa pemrograman *python*, terlebih dahulu bahasa pemrograman ini dianalisis bagaimana proses *encode* dan *decode*-nya bekerja. Kode yang akan dianalisa di kode BCH merupakan kode biner. Pada *encode* kode BCH, sebuah kode akan di-*encode* menjadi sebuah kode biner, lalu pada *decode,* kode BCH tersebut akan diubah kembali menjadi sebuah kode asli lalu dikoreksi menggunakan algoritma kode BCH. Selanjutnya, akan dianalisa algoritma serta programnya di dalam bahasa pemrograman *python.* Didapatkan hasil algoritma dan program *python* untuk mengoreksi sebuah kode di dalam kode BCH.

Keyword. Kode BCH, python, encode, decode, eror

1. Pendahuluan

Teori koding adalah ilmu yang mempelajari metode transmisi data melalui saluran komunikasi yang tidak bebas gangguan secara efisien dan akurat. Teori koding sangat diperlukan pada proses transmisi data dan penyimpanan data. Pada prosesnya, sering terjadi eror yang diakibatkan oleh adanya *noise* yang muncul dari proses transmisi sehingga data yang diterima tidak sesuai seperti yang dikirim dan diperlukan teori koding untuk membantu mendeteksi dan mengoreksi eror tersebut. Bidang teori koding terfokus pada ilmu yang memelajari tentang informasi, baik dalam pengamanan dan pengompresiannya. Pada proses pengamanan informasi tersebut, terjadi proses pendeteksian dan pengoreksian eror [7].

Sebelum lebih lanjut membahas terkait pendeteksian dan pengoreksian eror, ada beberapa hal yang perlu dibahas terkait teori koding yaitu perlunya dipelajari tentang sistem komunikasi. Sistem komunikasi pada teori koding dibagi menjadi dua proses. Proses pertama adalah sumber pesan akan melalui *encode* yaitu proses perubahan pesan menjadi sebuah *codeword* menggunakan sumber *encoder*-nya lalu menjadi sebuah input ke *channel.* Pada *channel* ini terdapat eror yang terjadi pada proses transmisi. Proses kedua adalah *codeword* yang sudah sampai ke *channel* tersebut akan dipecahkan menjadi pesan asli melalui *decode* dengan sumber *decoder*-nya lalu akan disampaikan ke tujuannya [8].

Selanjutnya akan dibahas terkait *channel coding*. Pengkodean kanal atau disebut dengan *channel coding* adalah pengkodean yang memiliki tujuan agar transmisi informasi pada sistem komunikasi menjadi lebih handal dan memiliki eror yang lebih kecil. *Channel coding* memiliki tahapan yaitu pendeteksian eror (*error detection*) dan pengoreksian eror (*error correcting*). Secara teknis, *channel coding* dilakukan dengan penambahan bit-bit *parity* yang artinya adalah bit yang ditambahkan untuk mendeteksi eror pada bit informasi dengan kode atau algoritma tertentu pada *decode*-nya.

Pendeteksian eror adalah mendeteksi apakah ada eror yang terjadi pada proses pertama di sistem komunikasi dengan output "ya" atau "tidak". Pengoreksian eror adalah mengoreksi eror yang terjadi pada proses transmisi dapat diperbaiki. Pada penelitian kali ini, penulis akan terfokus pada pengoreksian eror. Salah satu kode untuk melakukan pendeteksian dan pengoreksian kode adalah kode BCH.

Menurut Arista dkk. [2], kode BCH atau kode Bose-Chaudhuri-Hocquenghem adalah salah satu jenis kode pengoreksi eror bertipe siklik yang dibangun menggunakan asas himpunan terbatas. Kode BCH sendiri dibuat oleh matematikawan Prancis bernama Alexis Hocquenghem pada tahun 1959 bersama dua matematikawan lain yaitu Raj Bose dan D.K. Ray-Chaudhuri pada tahun 1960. BCH diambil dari singkatan ketiga ilmuwan tersebut. Pada proposal penelitian kali ini, akan dibahas bagaimana *encode* dan *decode* diimplementasikan pada kode BCH menggunakan bahasa pemrograman *python* untuk dilihat hasil koreksi eror yang digunakan dalam kode BCH. Tujuan dari penelitian ini untuk mengetahui bagaimana cara kerja dari kode BCH dan bagaimana implementasinya jika diimplementasikan dengan *pyton.* Untuk menggunakan kode BCH, perlu menganalisis beberapa parameter yang dibutuhkan lalu mencari polinomial generatornya, selanjutnya mencari eror yang muncul pada *encode*-nya. Alasan dipilihnya program *python* adalah bahasa *python* merupakan salah satu bahasa pemrograman yang mudah untuk dipelajari dan memiliki tingkat efesiensi tinggi pada struktur data serta memiliki fitur pembantu untuk program yang tinggi. Diharapkan dengan adanya program *python* untuk kode BCH ini dapat memudahkan untuk memahami bagaimana kode BCH bekerja.

2. Metode Penelitian

Permasalahan yang ada pada teori koding adalah bagaimana melakukan *encode* dan *decode* pada sebuah *channel* dan sebuah kode dan bagaimana eror yang muncul saat proses transmisi. Untuk mencari hal tersebut, pada penelitian kali ini akan dibuat algoritma dan program dengan menggunakan bahasa pemrograman *python* agar dapat mudah dipahami dan melakukan pendekatan terhadap kode tersebut.

Penelitian ini merupakan penelitian studi literatur. Studi literatur adalah rangkaian kegiatan yang berkenaan dengan metode pengumpulan data pustaka, membaca, mencatat, serta mengelola bahan penelitian [4]. Penelitian ini mencari teori dan studi terkait implementasi *encode* dan *decode* pada kode BCH. Langkah penelitian akan ditunjukkan pada Gambar 1.

3. Hasil Penelitian

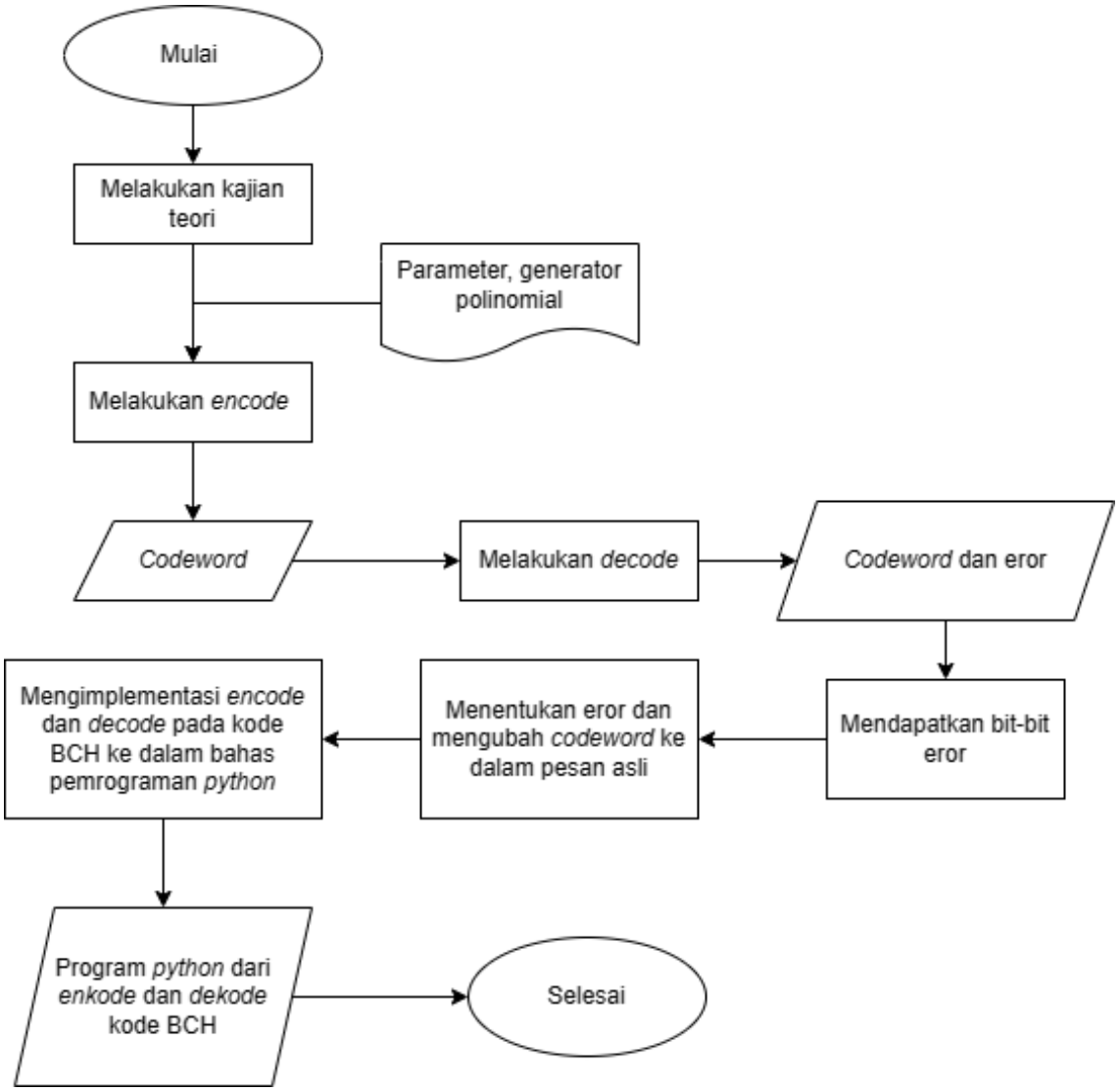
Kode BCH (Bose-Chaudhuri-Hocquenghem) telah ditemukan pada tahun 1959-1960. Kode ini merupakan kode bertipe siklik untuk mendeteksi dan mengoreksi eror. Kode BCH dapat mengoreksi eror ganda. Kode ini memiliki parameter n, k, t dan m dengan n adalah panjang kode ($n = 2^m - 1$), k adalah panjang pesan yang memenuhi ($k \geq n - mt$) , *t* adalah jumlah eror yang dapat diperbaiki dan *m* adalah nilai positif integer dimana ($m \geq 3$). Perhitungan menggunakan kode BCH dimulai dengan *encode* nya terlebih dahulu untuk mengubah pesan menjadi bentuk kode, lalu akan diproses melalui *decode*-nya untuk mengubah kode tersebut lalu dideteksi dan dikoreksi eror yang ada.

3.1. Encode

Proses *encode* yaitu proses mengubah pesan asli (*plain text*) menjadi sebuah *codeword*, proses-prosesnya tersebut antara lain:

i. Mencari parameter-parameter yang diperlukan.
ii. Membentuk *Galois Field* $GF(2^m)$.
iii. Menentukan $2t - 1$ buah minimal polynomial $m_n(x)$ dan ambil polynomial pangkat ganjil saja (polinomial pangkat genap merupakan duplikasi polinomial pangkat ganjil).
iv. Membentuk generator polinomial $G(x)$ kode BCH yang terbentuk dari
$$G(x) = KPK(m_1(x), m_3(x), m_5(x), \dots, m_{2t-1}(x))$$
v. Menambahkan bit di belakang bit pesan dengan panjang sebesar derajar generator polinomialnya.
vi. Melakukan operasi pembagian biner terhadap gabungan pesan sebagai *checkbit*.

vii. Melakukan operasi penambahan pesan informasi dan *checkbit* sebagai pesan yang akan dikirimkan sebagai kode ($V(x)$).



Gambar 1. Tahapan penelitian

3.2. *Decode*

Proses *decode* pada kode BCH adalah mendeteksi dan mengoreksi eror yang ada, prosedurnya adalah sebagai berikut.

a. *Error Detection*
   Membagi pesan yang dikirimkan sebagai kode dengan generator polinomialnya.
   i. Jika sisa pembagian $= 0$, maka tidak terjadi eror.
   ii. Jika sisa pembagian $\neq 0$, maka terjadi eror dan masuk ke prosedur *error correction.*
b. *Error Correction*
   i. Menentukan $2t$ buah minimal polinomial.
   ii. Menghitung sindrom $S_1, S_2, \dots, S_{2t}$ dengan $S_n = V(x)\ mod\ m_n(x)$
   iii. Mencari *bitflips* untuk menentukan dimana letak eror dan mengoreksinya.

3.3. Algoritma *Python*

Dalam mengimplementasikan *encode* dan *decode* dari kode BCH tersebut, untuk menyusun algoritma *python* digunakan module *python bchlib* untuk membantu prosesnya. *Python* yang digunakan adalah *Python 3*. Berikut adalah kode yang digunakan.

```
import bchlib
import hashlib
import os
import random
```

Gambar 2. Meng-impor *module bchlib*.

```
BCH_POLYNOMIAL = 8219
BCH_BITS = 16
bch = bchlib.BCH(BCH_POLYNOMIAL, BCH_BITS)

data = bytearray(os.urandom(127))
```

Gambar 3. Menentukan parameter.

```
ecc = bch.encode(data)
packet = data + ecc

sha1_initial = hashlib.sha1(packet)
print('sha1: %s' % (sha1_initial.hexdigest(),))

def bitflip(packet):
    byte_num = random.randint(0, len(packet) - 1)
    bit_num = random.randint(0, 7)
    packet[byte_num] ^= (1 << bit_num)

for _ in range(BCH_BITS):
    bitflip(packet)

sha1_corrupt = hashlib.sha1(packet)
print('sha1: %s' % (sha1_corrupt.hexdigest(),))
```

Gambar 4. Melakukan *encode*.

```
data, ecc = packet[:-bch.ecc_bytes], packet[-bch.ecc_bytes:]

bitflips = bch.decode_inplace(data, ecc)
print('bitflips: %d' % (bitflips))

packet = data + ecc

sha1_corrected = hashlib.sha1(packet)
print('sha1: %s' % (sha1_corrected.hexdigest(),))

if sha1_initial.digest() == sha1_corrected.digest():
    print('Corrected!')
else:
    print('Failed')
```

Gambar 5. Melakukan *decode*.

Lalu didapatkan output sebagai berikut.

```
sha1: 8a3429ddaa60f34c8c7615969b602729289517bc
sha1: 4d1b4b4eed7f54de2ed350a23c608842d55803f1
bitflips: 16
sha1: 8a3429ddaa60f34c8c7615969b602729289517bc
Corrected!
```

Gambar 6. Output dari kode tersebut yang artinya parameter tersebut dapat dikoreksi dan selesai dikoreksi.

4. Kesimpulan dan Saran

Setelah dilakukan uji coba terhadap program *python* nya dan kode dengan parameter dapat dikoreksi. Kode BCH tersebut dapat diimplementasikan ke dalam program *python* dengan menggunakan bantuan dari *module bchlib* dengan program sebagai berikut.

```python
import bchlib
import hashlib
import os
import random

BCH_POLYNOMIAL = 8219
BCH_BITS = 16
bch = bchlib.BCH(BCH_POLYNOMIAL, BCH_BITS)

data = l

ecc = bch.encode(data)
packet = data + ecc

sha1_initial = hashlib.sha1(packet)
print('sha1: %s' % (sha1_initial.hexdigest(),))

def bitflip(packet):
    byte_num = random.randint(0, len(packet) - 1)
    bit_num = random.randint(0, 7)
    packet[byte_num] ^= (1 << bit_num)

for _ in range(BCH_BITS):
    bitflip(packet)

sha1_corrupt = hashlib.sha1(packet)
print('sha1: %s' % (sha1_corrupt.hexdigest(),))

data, ecc = packet[:-bch.ecc_bytes], packet[-bch.ecc_bytes:]

bitflips = bch.decode_inplace(data, ecc)
print('bitflips: %d' % (bitflips))

packet = data + ecc

sha1_corrected = hashlib.sha1(packet)
print('sha1: %s' % (sha1_corrected.hexdigest(),))

if sha1_initial.digest() == sha1_corrected.digest():
    print('Corrected!')
else:
    print('Failed')
```

Gambar 7. Program yang digunakan untuk implementasi kode BCH ke dalam program *python*

Saran yang dapat diambil adalah penulis dapat mengembangkan penelitian dengan melakukan uji coba menggunakan parameter yang besar nilainya dan membuat ulang kode *python* tanpa menggunakan *module.*

5. Daftar Pustaka

[1] Amin, M.M., 2016. Implementasi Kriptografi Klasik pada Komunikasi Berbasis Teks. Pseudocode, 3(2), pp.129-136.
[2] Arista, L., Dhoruri, A., and Lestari, D., 2016. Encoding dan Decoding Kode BCH (Bose Chaudhuri Hocquenghem) Untuk Transmisi Data.
[3] Deswanti, F.M., Hidayat, B., and Aulia, S., 2015. Steganografi Citra Digital Menggunakan Enkripsi Berdasarkan Prinsip Kubus Rubik Dan Kode BCH.
[4] Kartiningrum, E.D., 2015. Panduan Penyusunan Studi Literatur. Lembaga Penelitian Dan Pengabdian Masyarakat Politeknik Kesehatan Majapahit, Mojokerto, pp.1-9.
[5] Mohammed, S.J., 2013. Implementation of Encoder for (31, k) Binary BCH Code based on FPGA for Multiple Error Correction Control. International Journal of Computer Applications, 76(11).
[6] Muhajir, F., 2016. Analisis Deteksi dan Koreksi Multi Bit Error dengan Partition Hamming Code.
[7] Roering, C., 2013. Coding Theory-based Cryptography: McEliece Cryptosystems in Sage.
[8] Roth, R.M., 2006. Introduction to Coding Theory. IET Communications, 47(18-19), p.4.
[9] Sutarto, M., Suwadi, S., and Suryani, T., 2014. Implementasi Encoder dan Decoder BCH Menggunakan DSK TMS320C6416T. Jurnal Teknik ITS, 3(1), pp.A29-A34.
[10] Yoo, H., Jung, J., Jo, J. and Park, I.C., 2013. Area-efficient Multimode Encoding Architecture for Long BCH Codes. IEEE Transactions on Circuits and Systems II: Express Briefs, 60(12), pp.872-876

Ucapan terima kasih

Alhamdulillah, telah terlaksana seminar hasil untuk penelitian ini, saya ingin mengucapkan rasa syukur kepada Allah SWT yang telah memberikan sehat dan kesempatan sehingga dapat menyelesaikan artikel serta telah melaksanakan seminar. Selanjutnya saya ingin mengucapkan terima kasih kepada orang tua saya, Ayah dan Umi yang telah memberikan dukungan finansial dan motivasi kepada saya. Selanjutnya saya ingin mengucapkan terima kasih kepada pembimbing saya Pak Putranto Hadi Utomo dan Pak Bowo Winarno yang terus membimbing saya sehingga dapat menyelesaikan artikel dan penelitian ini. Tidak lupa saya ucapkan terima kasih kepada teman-teman saya yang terus memberikan dorongan positif kepada saya.