SIMULASI PENYANDI DAN PENGAWASANDI SANDI ORTHOGONAL 8 BIT DENGAN MATLAB

TUGAS AKHIR

Diajukan untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Teknik
Program Studi Teknik Elektro

Oleh :

CAHYO INDRATOMO

NIM : 015114075

PROGRAM STUDI TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS SANATA DHARMA
YOGYAKARTA

2006

8 BIT ORTHOGONAL ENCODER AND DECODER CODE SIMULATION USING MATLAB

A FINAL PROJECT

Submitted For The Partial Fulfillment Of The Requirements
For The Degree Of Electrical Engineering Program Study

By :

CAHYO INDRATOMO

NIM : 015114075

ELECTRICAL ENGINEERING STUDY PROGRAM
FACULTY OF ENGINEERING
SANATA DHARMA UNIVERSITY
YOGYAKARTA
2006

HALAMAN PERSETUJUAN

TUGAS AKHIR

SIMULASI PENYANDI DAN PENGAWASANDI SANDI ORTHOGONAL 8 BIT DENGAN MATLAB

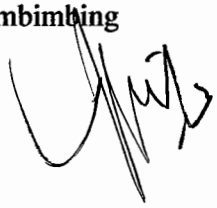
Oleh :

CAHYO INDRATOMO

NIM : 015114075

Telah disetujui oleh

Pembimbing



Ir. Th. Prima Ari Setiyani, MT

Tanggal 15 Desember 2006

HALAMAN PENGESAHAN

TUGAS AKHIR

SIMULASI PENYANDI DAN PENGAWASANDI SANDI ORTHOGONAL 8 BIT DENGAN MATLAB

Disusun oleh :

CAHYO INDRATOMO

NIM : 015114075

Telah dipertahankan di hadapan Panitia Penguji

Pada tanggal : 20 Desember 2006

dan dinyatakan telah memenuhi syarat

Susunan Panitia Penguji :

		Nama Lengkap	Tanda Tangan
Ketua	:	Wiwien Widyastuti, S.T., M.T	
Sekretaris	:	Ir. Th. Prima Ari Setiyani, M.T.	
Anggota	:	Martanto, S.T, M.T.	
Anggota	:	Damar Widjaja, S.T, M.T	

Yogyakarta, 15 Januari 2007

Fakultas Teknik

Universitas Sanata Dharma

Dekan,

Ir. Greg. Heliarko S.J.S.S.,B.S.T.,M.A.,M.Sc.

HALAMAN MOTO DAN PERSEMBAHAN

- *Tiada pekerjaan yang paling baik kecuali pekerjaan yang dilakukan dengan usaha (tangan sendiri) dan perdagangan bersih (HR. Al Hakim).*
- *Setiap kamu adalah pemimpin dan setiap pemimpin akan dimintai pertanggung jawaban (HR Bukhari- Muslim).*
- *Orang berilmu senantiasa abadi dalam ingatan meski tulang belulangnya hancur dimakan tanah.*

  Dan orang yang tidak berpengetahuan seolah jasad tak bernyawa bagaikan orang yang hidup dalam kematian.
- *Ilmu pengetahuan menghidupkan hati yang mati sebagaimana hujan menyirami bumi yang tandus.*

  Ilmu pengetahuan menyinari kegelapan kalbu dan seakan purnama menerangi gulita malam.
- *Seseorang dengan tujuan yang jelas akan mmembuat kemajuan walaupun melewati jalan yang sulit. Seseorang yang tanpa tujuan, tidak akan membuat kemajuan walaupun ia berada di jalan yang mulus! (Thomas Carlyle)*

Kupersembahkan Tugas Akhir ini kepada
Allah SWT atas Rahmat, Hidayah dan Inayah-Nya
Bapakku M. Pratomo dan Ibuku Nur'aini yang tercinta
Nur Kholifah yang terkasih

PERNYATAAN KEASLIAN KARYA

Saya menyatakan dengan sesungguhnya bahwa tugas akhir ini tidak memuat karya atau bagian karya orang lain, kecuali yang telah disebutkan dalam kutipan dan daftar pustaka sebagaimana layaknya karya ilmiah.

Yogyakarta, Januari 2007

Penulis

Cahyo Indratomo

SIMULASI PENYANDI DAN PENGAWASANDI SANDI ORTHOGONAL 8 BIT DENGAN MATLAB

INTISARI

Salah satu tujuan dalam transmisi data adalah kesamaan atau keselarasan antara data yang dikirim oleh pengirim dan data yang diterima oleh penerima. Tetapi dalam beberapa keadaan, hal seperti ini sulit dicapai. Terkadang data yang diterima oleh penerima tidak sesuai dengan data yang dikirim oleh pengirim. Sandi orthogonal dapat mendeteksi *error* baik ganjil maupun genap serta mengkoreksinya. Dalam tugas akhir ini penulis akan membuat simulasi pembangkit CODEC Orthogonal 8 bit dengan deteksi *error* dan koreksi *error* menggunakan MATLAB.

Sandi orthogonal 8 bit dapat diubah menjadi katasandi yang masing-masing terdiri dari 256 bit. Setiap katasandi dalam matrik yang dibandingkan dengan kata sandi lain mempunyai digit yang sama sebanyak digit yang tidak sama. Setelah proses penyandian, pengguna dapat merubah kata sandi dengan mengubah satu bit atau lebih dalam kata sandi tersebut. Kemudian kata terima tersebut dideteksi. Pada proses deteksi *error*, kata terima dibandingkan dengan salah satu sandi dari *lookup table* untuk mendapatkan nilai Z_{ij}. Pada proses *lookup table*, jika berhasil berarti tidak ada *error*. Tapi jika *lookup table* gagal berarti ada *error*. Apabila ada *error*, maka akan dicari nilai dmin (jarak minimum dari sandi), kemudian dilakukan proses *lookup table* yang akan menghasilkan data terkirim 8 bit.

Program ini mampu menyandikan data 8 bit menjadi katasandi 256 bit dengan benar. Proses deteksi dilakukan dengan cara menghitung nilai Z_{ij}. Nilai Z_{ij} digunakan untuk mengetahui sandi tersebut orthogonal atau tidak. Pada proses *lookup table*, jika proses *lookup* berhasil, maka program akan mendeteksi bahwa tidak ada *error* pada kata terima. Tapi jika *lookup table* gagal berarti ada *error* pada kata terima.

8 BIT ORTHOGONAL ENCODER AND DECODER CODE SIMULATION USING MATLAB

ABSTRACT

One of the goals in data transmission is equality or compatibility between data sent by transmitter and data that is accepted by receiver. But in some circumstance, this condition is hard to be archived. Sometime data that is accepted by receiver is not the same as the data that is sent by transmitter. Orthogonal Code can detect the even and also anomalous good error and also correct it. In this final duty is writer will make the simulation of generating of Orthogonal Code 8 bit detected error and correct the error use MATLAB.

The orthogonal code 8 bit can be change into codeword which is each consisting of 256 bit. Every codeword in matrix if comparisons with other code have the same digit as much unequal digit. After process encoding, consumer can change the password by altering one bit or more of the password. Then accept word detected. At process detect error, accept word compare with something codes from lookup table to get the value Z_{ij}. At lookup table process, if succeed that means no error. But if lookup table failed means error. When ever is error, therefore will looks dmin value (apart the minimum from code), then conducted process of lookup table to yield the 8 bit data sent.

This program able to encode 8 bit data becomes the code word 256 bit truly. Process detects conducted by calculating Zij value. Zij value used for to know that orthogonal code or no. At lookup table process, if lookup table process succeed, therefore program will detect that there no error. But if lookup table failed means is error at accepted word.

KATA PENGANTAR

Alhamdulillah puji dan syukur tiada pernah henti penulis panjatkan kehadirat Allah SWT, zat yang telah melimpahkan rahmat, hidayah dan inayah-Nya. Shalawat serta salam tidak lupa penulis haturkan kepada Nabi Besar Muhammad SAW, sehingga penulis dapat menyelesaikan tugas akhir ini yang berjudul "Simulasi Penyandi dan Pengawasandi Sandi Orthogonal 8Bit dengan Matlab".

Penyusunan tugas akhir ini sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik di Fakultas Teknik, Universitas Sanata Dharma, Yogyakarta. Penyusunan tugas akhir ini tidak akan terwujud tanpa adanya bantuan dan dukungan dari berbagai pihak, baik berupa pengarahan maupun bantuan yang sangat berguna bagi penulis. Dengan terselesaikannya penyusunan tugas akhir ini, maka penulis tidak akan melupakan semua kebaikan dan jasa-jasanya, oleh karena itu pada kesempatan yang baik ini, penulis menyampaikan rasa hormat dan terima kasih kepada:

1. Ir. Th. Prima Ari Setiyani, M.T. sebagai pembimbing Tugas Akhir atas waktu, bimbingan, nasehat, kesabaran, pengarahan, koreksi dan saran yang diberikan selama penyusunan tugas akhir.
2. Ir. Greg. Heliarko, S.J., S.S., B.S.T., M.A., M.Sc. selaku Dekan Fakultas Teknik.
3. Bapak dan Ibu dosen yang telah banyak memberikan pengetahuan dan bimbingan kepada penulis selama menempuh kuliah.
4. Bapak Djito, Mas Suryono, Mas Mardi, Mas Broto dan Pak Hardi yang sangat membantu dalam penyelesaian administrasi dan sarana laboratorium.
5. Bapakku M. Pratomo dan Ibuku Nur'aini tercinta yang dengan keikhlasan hati telah memberikan doa, nasehat, dukungan, perhatian, dorongan semangat, kasih sayang dan kesabarannya serta materi kepada penulis sehingga tugas akhir ini dapat terselesaikan dengan baik.

6. Mbah Soeharto, mbah Siti Chatidjah, mbah Sosrowijoyo, budhe Titi Pratiwi, mbak Asri Agustina Setyowati yang telah banyak membantu aku selama menempuh pendidikan di jogja ini. Terimakasih atas perhatian dan nasehat serta doanya selama ini.
7. Mbak Tatik, mbak Ambar, mbak Ari, adikku Utami, adikku Ira, adikku Tanto terimakasih atas perhatian dan doanya selama ini.
8. Nur "Ifa" Kholifah atas kasih sayang, dukungan dan doanya, U'r my inspiration.
9. Sobatku Jhon Kiegen Maksimillanus Sinaga "Abang", Jack 9npar, Indarto, Tyo "su-uk", tatank serta teman-teman TE 2001 yang tidak bisa penulis sebutkan satu per satu, terima kasih atas dukungannya.
10. Perpustakaan dan kepada semua pihak yang tidak dapat penulis sebutkan satu persatu atas segala bantuan dan doanya.

Akhir kata Penulis berharap semoga tugas akhir ini dapat bermanfaat bagi kita semua.

Yogyakarta, Januari 2007

Penulis

Cahyo Indratomo

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Halaman

BAB I

PENDAHULUAN

1.1. Latar Belakang Masalah

Salah satu tujuan dalam transmisi data adalah kesamaan atau keselarasan antara data yang dikirim oleh pengirim dan data yang diterima oleh penerima. Tetapi dalam beberapa keadaan hal seperti ini sulit dicapai. Terkadang data yang diterima oleh penerima tidak sesuai dengan data yang dikirim oleh pengirim. Bila hal itu terjadi, maka harus dicari suatu metode yang dapat mendeteksi *error* tersebut, sehingga data yang diterima dapat diketahui benar atau tidak. Jika masih terdapat *error*, maka penerima akan meminta pengiriman data ulang atau melakukan koreksi sendiri.

Dalam mendeteksi *error*, dapat menggunakan paritas ganjil, paritas genap, *orthogonal codec*, dan lain sebagainya. Pada deteksi *error* menggunakan Sandi uji paritas tunggal hanya dapat mendeteksi jumlah *error* yang ganjil (baik paritas ganjil maupun paritas genap). Dengan kata lain, sandi ini tidak dapat mendeteksi jumlah *error* yang genap. Tetapi dengan menggunakan metode orthogonal dapat mendeteksi *error* baik ganjil maupun genap.

Dalam tugas akhir ini penulis akan membuat simulasi pembangkit CODEC Orthogonal 8 bit dengan deteksi *error* dan koreksi *error* menggunakan MATLAB.

1.2. Batasan Masalah

Dalam penulisan tugas akhir ini penulis akan membuat deteksi *error* dari kata terima dengan menggunakan bantuan *software* MATLAB dengan batasan :

a. Membuat simulasi sandi Orthogonal 8 bit
b. Membuat fasilitas untuk membuat kata sandi menjadi *error*
c. Mendeteksi ada tidaknya *error*

1.3. Metode Penelitian

Metode yang dipakai dalam penyusunan tugas akhir ini adalah dengan melakukan:

a. Studi literatur dengan mempelajari buku referensi dan mencari data yang berkaitan dengan pembahasan tugas akhir.
b. Pembuatan fasilitas pembuat *error*, pendeteksi dan pengkoreksi *error* dengan menggunakan MATLAB.

1.4. Tujuan dan Manfaat Penelitian

Tujuan yang akan dicapai dalam penelitian ini adalah menghasilkan suatu program bantu untuk mendeteksi *error*.

Manfaat yang dicapai dalam melakukan penelitian ini adalah mendalami ilmu pengetahuan tentang sandi Orthogonal, serta dapat juga untuk alat bantu dosen untuk proses belajar-mengajar.

1.5. Sistematika Penulisan

Proposal Tugas Akhir ini ditulis dengan sistematika sebagai berikut :

BAB I PENDAHULUAN

Bab I berisi tentang judul, latar belakang, perumusan masalah, batasan masalah, tujuan dan manfaat penelitian dan sistematika penulisan

BAB II DASAR TEORI

Dasar teori akan menjelaskan tentang metoda pembuatan sandi Orthogonal.

BAB III PERANCANGAN PROGRAM

Bab III akan menjelaskan perancangan *software* dengan menggunakan diagram alir (*Flow Chart*) serta penjelasan singkat tentang cara kerja *encoder-decoder* pendeteksi *error* sandi Orthogonal 8 bit.

BAB IV ANALISA DAN PEMBAHASAN

Bab IV akan dijelaskan tentang cara kerja dan analisa program *encoder-decoder* pendeteksi *error* sandi Orthogonal 8 bit.

BAB V PENUTUP

Bab V memuat kesimpulan, kritik dan saran.

REFERENSI

BAB II
LANDASAN TEORI

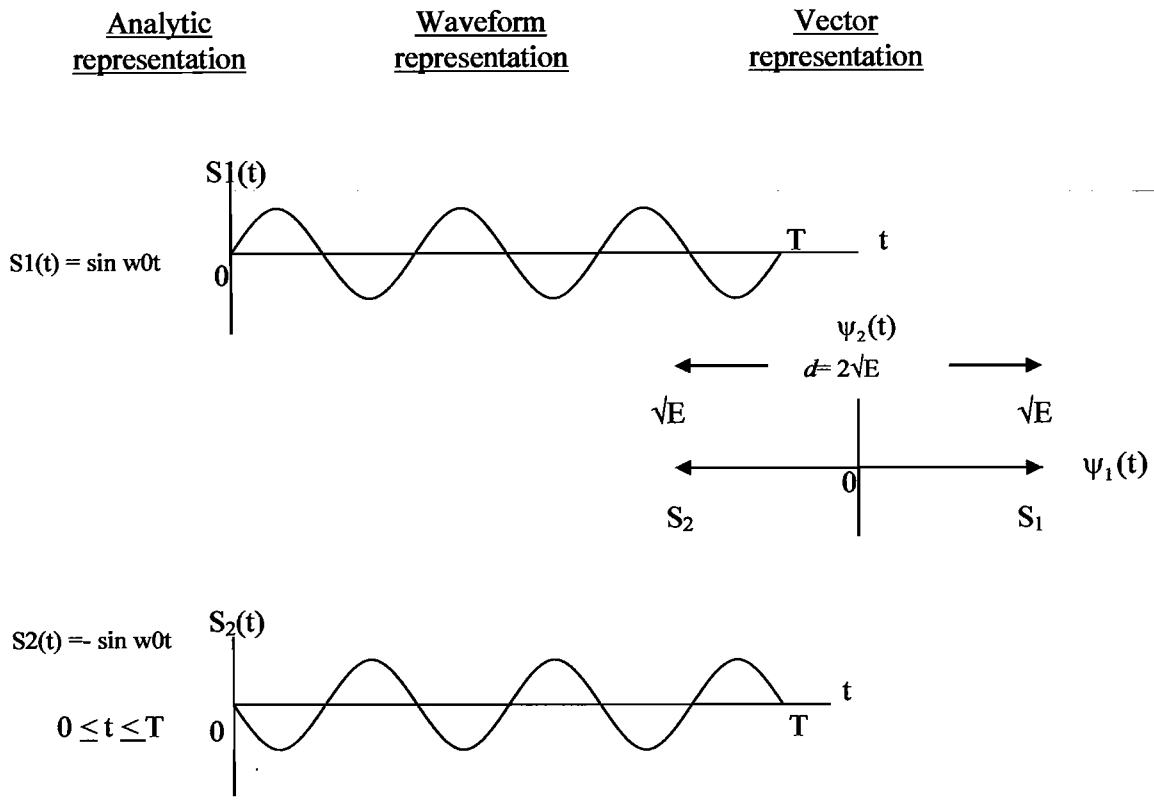
2.1. Penyandian Isyarat Digital

Penyandian isyarat digital berguna untuk meningkatkan kinerja komunikasi dengan meningkatkan ketahanan sinyal terhadap berbagai gangguan saluran, seperti derau (*noise*), pelemahan sinyal (*fading*), dan perusakan sinyal oleh sinyal lain (*jamming*). Penyandian dikelompokkan dalam dua jenis yaitu penyandian gelombang (*waveform coding*) dan urutan terstruktur (*stuctured sequences). Waveform coding* berhubungan dengan pengubahan bentuk gelombang menjadi bentuk gelombang yang "lebih baik", sehingga mengurangi terjadinya galat (*error*). Sedangkan *stuctured sequences* berhubungan dengan pengubahan urutan data menjadi urutan yang "lebih baik" dan mempunyai bit berlebih (*redundant bit*), yang digunakan untuk mendeteksi dan mengkoreksi *error*.

2.1.1 Penyandian Gelombang (*Waveform Coding*)

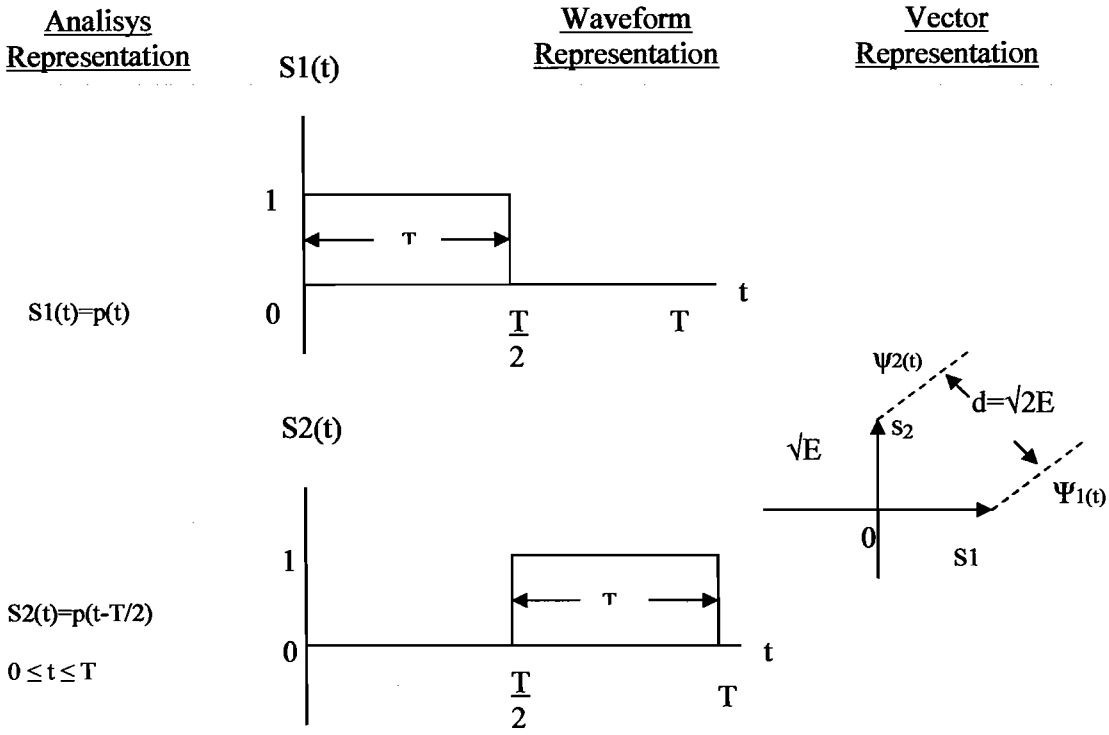
2.1.1.1 Sinyal Antipodal dan Sinyal Orthogonal

Sinyal antipodal adalah negatif dari sinyal yang lainnya. Dengan kata lain, sinyal antipodal adalah cermin dari sinyal lain. Dalam hal sinyal sinusoida, maka sepasang sinyal antipodal adalah sinyal-sinyal yang berbeda fasa 180°. Gambar 2.1 memperlihatkan sepasang sinyal antipodal sinusoida.



Gambar 2.1 Contoh sinyal antipodal

Contoh sinyal orthogonal diperlihatkan pada gambar 2.2. Dari gambar terlihat bahwa sinyal $s_1(t)$ menempati area waktu yang berbeda dengan $s_2(t)$, $p(t)$ merupakan fungsi pulsa dengan durasi $\tau = T/2$. Sinyal pada gambar 2.2 tidak saling mengganggu, karena tidak muncul dalam waktu yang sama. Sinyal dengan fungsi sinusoida, sin x dan cos x berbeda fasa sebesar 90°, oleh karena itu sin x dan cos x merupakan sinyal orthogonal. Demikian juga sin mx dan sin nx, dengan m dan n adalah bilangan bulat, dan $m \neq n$.



Gambar 2.2 Contoh sinyal orthogonal biner

Secara umum, satu set sinyal energi yang sama $s_i(t)$, dengan $i = 1,2,....,$ M, dikatakan orthogonal, jika dan hanya jika,

$$Z_{ij} = \frac{1}{E}\int_0^T s_i(t)\, s_j(t)dt = \begin{cases} 1 & \text{untuk } i = j \\ 0 & \text{untuk } i \neq j \end{cases} \qquad (2.1)$$

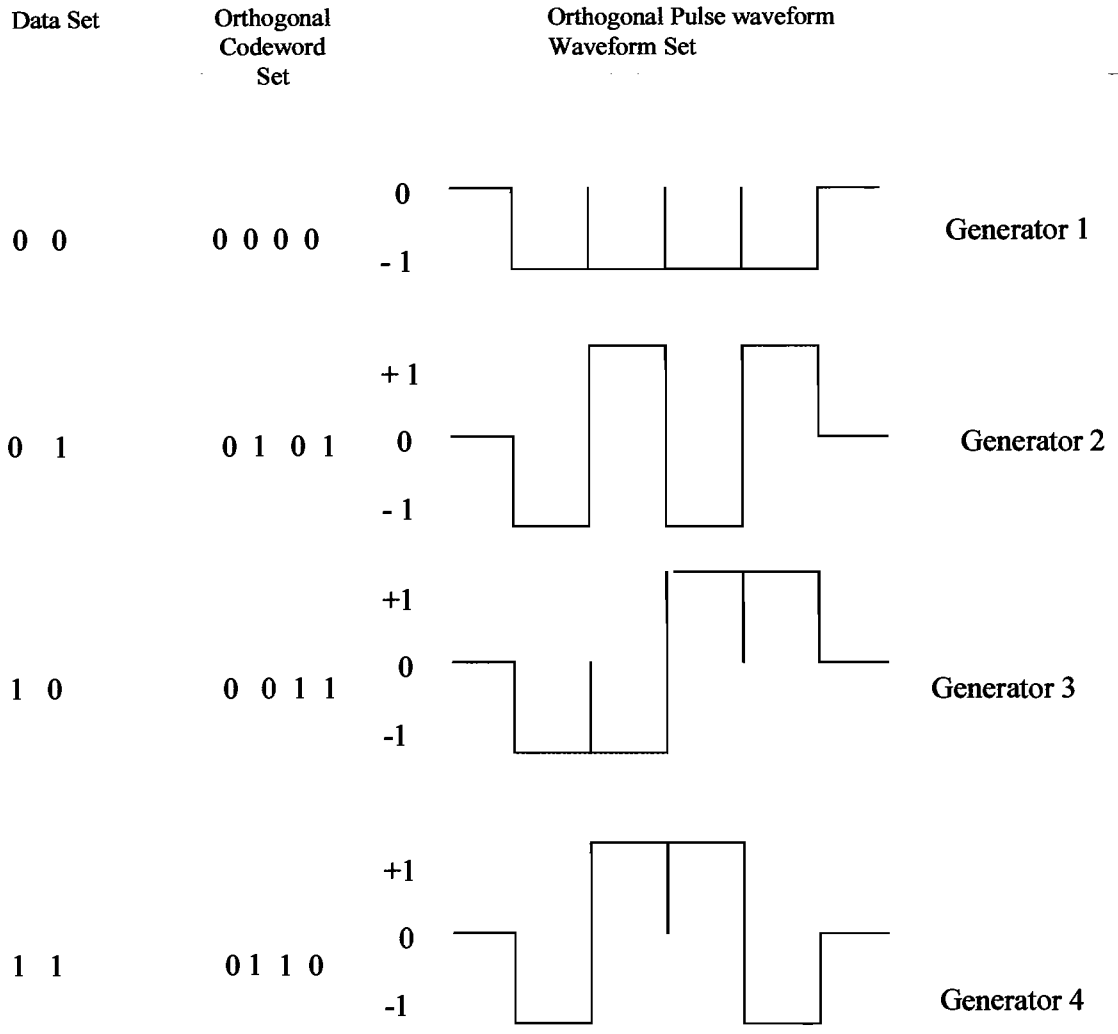
Z_{ij} adalah koefisien korelasi silang (*cross correlation coefficient*), dan E adalah energi sinyal.

2.1.2. Penyandian Gelombang dengan deteksi korelasi

Penyandian gelombang mengubah satu set gelombang menjadi satu set gelombang yang lebih baik. Pengubahan paling populer adalah dengan sandi orthogonal dan sandi biorthogonal. Tujuan dari prosedur penyandian ini adalah untuk membuat setiap bentuk gelombang sinyal sebeda mungkin, atau membuat z_{ij} dari setiap pasangan sinyal sekecil mungkin. Nilai terkecil dari z_{ij} terjadi jika satu sinyal merupakan antikorelasi dari sinyal yang lain (z_{ij} = -1). Tetapi hal ini dapat tercapai jika jumlah simbol dalam kelompok sinyal adalah dua (M=2) dan setiap simbol adalah antipodal. Secara umum z_{ij} dapat dibuat sama dengan nol, dan kelompok sinyal itu disebut orthogonal. Satu set sinyal antipodal adalah optimum, jika setiap sinyal berada pada jarak terjauh dengan sinyal yang lain. Pada gambar 2.1, jarak d antara vektor sinyal adalah $d = 2\sqrt{E}$, dengan E adalah energi sinyal selama durasi T. Satu set sinyal orthogonal pada gambar 2.2 mempunyai jarak $d = \sqrt{(2E)}$.

Korelasi silang antara dua sinyal merupakan ukuran jarak antara vektor-vektor sinyal. Semakin kecil korelasi silang, jarak antar vektor semakin panjang. Dari gambar 2.1, terlihat bahwa sinyal-sinyal antipodal (dengan z_{ij} = -1) digambarkan sebagai vektor dengan jarak terjauhnya (arah berlawanan), sedangkan pada gambar 2.2 sinyal orthogonal (dengan z_{ij} = 0) digambarkan sebagai vektor dengan jarak lebih dekat. Jarak antar sinyal yang identik (dengan z_{ij} = 1) adalah nol.

Gambar 2.3 memperlihatkan satu set data 2-bit yang diubah menjadi kelompok kata sandi orthogonal. Data asli dan kata sandi pengganti terdiri dari digit biner (1, 0). Terlihat juga gelombang yang terdiri dari pulsa bipolar (+1, -1) yang menggambarkan kata sandi.



Gambar 2.3 Perubahan satu set data dengan kata sandi orthogonal dan gelombang bipolar

Persamaan (2.1) ditulis dalam bentuk gelombang, tetapi jika kelompok sinyal diwakili oleh digit biner, maka persamaan tersebut dapat disederhanakan menjadi :

$$z_{ij} = \frac{jumlah\ digit\ yang\ sama - jumlah\ digit\ yang\ tidak\ sama}{jumlah\ total\ digit} \quad (2.2)$$

$$z_{ij} = \begin{cases} 1 & \text{untuk } i = j \\ 0 & \text{untuk } i \neq j \end{cases}$$

dengan i, j = 1, 2, ..., M, dan M adalah ukuran kata sandi. Pengiriman data dengan penggantian data dengan kata sandi orthogonal membuat jarak antar sinyal tersandi menjadi semakin jauh.

2.1.3 Penyandian Sandi Orthogonal

Data 1-bit dapat diubah menjadi kata sandi orthogonal yang masing-masing terdiri dari 2-bit. Hal ini dapat dijelaskan dengan matriks H_1 sebagai berikut :

Data	Katasandi Orthogonal
0	$H_1 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$
1	

Untuk menyandikan data 2-bit, maka data sebelumnya diperluas secara horisontal dan vertikal, sehingga menghasilkan H_2 sebagai berikut :

Data	Katasandi Orthogonal
0 0	$H_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} H_1 & H_1 \\ H_1 & \overline{H_1} \end{bmatrix}$
0 1	
1 0	
1 1	

Kuadran kanan bawah adalah komplemen dari katasandi sebelumnya. Dengan cara yang sama, dapat dihasilkan katasandi orthogonal untuk data 3-bit sebagai berikut :

Data	Katasandi Orthogonal
0 0 0	0 0 0 0 | 0 0 0 0
0 0 1	0 1 0 1 | 0 1 0 1
0 1 0	0 0 1 1 | 0 0 1 1
0 1 1	0 1 1 0 | 0 1 1 0
1 0 0	0 0 0 0 | 1 1 1 1
1 0 1	0 1 0 1 | 1 0 1 0
1 1 0	0 0 1 1 | 1 1 0 0
1 1 1	0 1 1 0 | 1 0 0 1

$$H_3 = \left(\begin{array}{cccc:cccc} 0&0&0&0&0&0&0&0 \\ 0&1&0&1&0&1&0&1 \\ 0&0&1&1&0&0&1&1 \\ 0&1&1&0&0&1&1&0 \\ \hdashline 0&0&0&0&1&1&1&1 \\ 0&1&0&1&1&0&1&0 \\ 0&0&1&1&1&1&0&0 \\ 0&1&1&0&1&0&0&1 \end{array}\right) = \begin{bmatrix} H_2 & H_2 \\ H_2 & \overline{H_2} \end{bmatrix}$$

Secara umum, matriks katasandi (*Hadamard Matrix*), H_k , dengan dimensi 2^k x 2^k untuk data *k*-bit dapat dibangun dari matriks H_{k-1} , sebagai berikut :

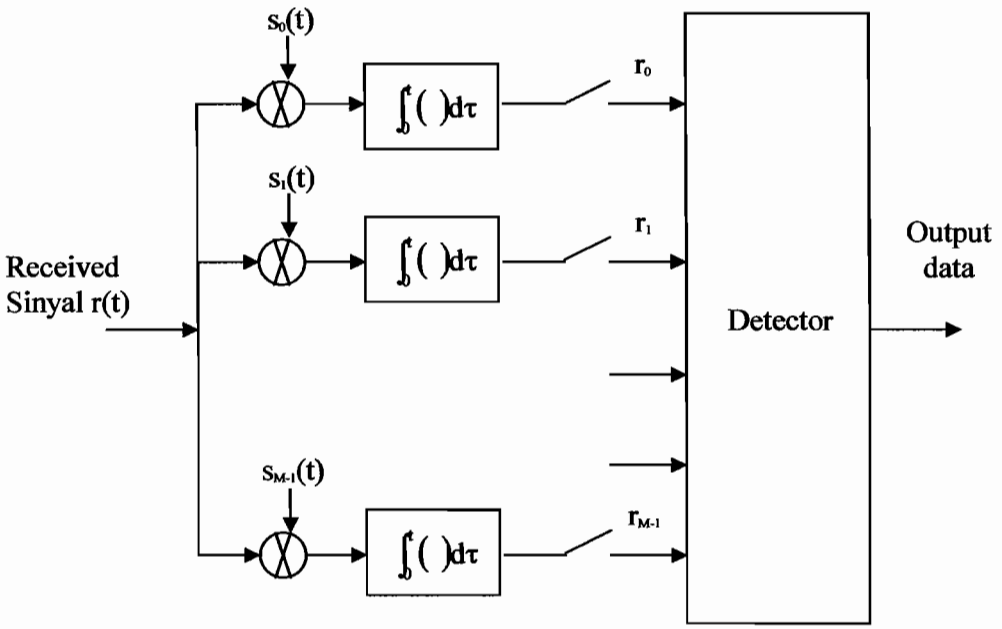
$$H_k = \begin{bmatrix} H_{k-1} & H_{k-1} \\ H_{k-1} & \overline{H_{k-1}} \end{bmatrix} \qquad (2.3)$$

Setiap katasandi dalam matriks jika dibandingkan dengan kata sandi lainnya (selain data 0) mempunyai digit yang sama sebanyak digit yang tidak sama atau jumlah digit 0 sama dengan jumlah digit 1, sehingga sesuai dengan persamaan (2.2), $z_{ij} = 0$ (untuk $i \neq j$).

Dengan asumsi bahwa bentuk gelombang orthogonal ini digunakan untuk sebuah informasi masukan suatu AWGN (*Additive White Gaussian Noise) Channel*, sehingga, jika bentuk gelombang yang dipancarkan $s_i(t)$, maka bentuk gelombang yang diterima adalah

$$R(t) = s_i(t) + n(t), \qquad 0 \le t \le T, \qquad i = 0,1, ..., M-1 \qquad (2.4)$$

n(t) adalah suatu fungsi contoh proses derau Gaussian dengan power spektrum No/2 (watts/hertz). Penerima mengamati dan memutuskan sinyal r(t), dengan M adalah bentuk gelombang isyarat yang dipancarkan. Penerima yang memperkecil kemungkinan kesalahan yang pertama lewat isyarat r(t) melalui suatu bank paralel M kemudian disaring dengan *matched filter* atau M *correlators*. Sejak dalam sinyal *correlators* dan bertemu saringan menghasilkan keluaran yang sama, seperti ditunjukkan di dalam gambar 2.4.



Gambar 2.4 Jumlah maksimum penerima untuk kode orthogonal

2.2. KEKUATAN PENGKODEAN

2.2.1. Bobot dan jarak dari vektor biner

Kemampuan untuk mengoreksi kesalahan suatu kode akan ditentukan oleh strukturnya. *Hamming weight* (bobot Hamming), *w*(U), dari suatu vektor U didefinisikan sebagai jumlah elemen *nonzero* di dalam U. Untuk suatu vektor

biner, ini adalah setara dengan jumlah bit 1 pada vektor itu. Sebagai contoh, jika U= 100101101, maka w(U) = 5. *Hamming distance* (jarak Hamming) antara dua vektor kode U dan V, ditulis d(U,V), didefinisikan sebagai banyaknya elemen yang berbeda, sebagai contoh :

U= 1 0 0 1 0 1 1 0 1
V= 0 1 1 1 1 0 1 0 0
d(U,V) = 6

Dengan sifat-sifat penambahan modulo-2, diketahui bahwa penjumlahan dua vektor biner adalah vektor yang lain dengan bit 1 yang terletak pada posisi di mana kedua vektor itu berbeda, sebagai contoh

U+V= 1 1 1 0 1 1 0 0 1

Dengan demikian *Hamming distance* antara dua vektor sandi sama dengan *Hamming weight* dari penjumlahan kedua vektor tersebut, d(U,V) = w(U+V). Terlihat juga bahwa *Hamming weight* suatu vektor sandi yaitu sama dengan *Hamming distance*-nya dari *all-zeros* vektor atau vektor dengan semua elemen bernilai nol.

2.2.2. Jarak minimum suatu kode linier

Andaikan sebuah himpunan yang anggota-anggotanya adalah jarak Hamming antar vektor sandi dan vektor *space* V_n, maka anggota paling kecil menjadi jarak minimum (*minimum distance)* dari sandi dan ditulis d_{min}. Jarak minimum, seperti mata rantai yang paling lemah di dalam suatu rantai, memberi kita suatu ukuran menyangkut kemampuan minimum kode dan kekuatan kode.

Penjumlahan dua vektor kode akan menghasilkan vektor kode lain yang tetap merupakan anggota *subspace*. Sifat kode linier tersebut dinyatakan secara sederhana sebagai : Jika U dan V adalah vektor kode, maka W = U + V harus pula

suatu vektor kode. Karenanya jarak antara dua vektor kode sama dengan *weight* vektor kode ketiga, didefinisikan sebagai berikut :

$$d(U,V)= w(U+V)= w(W) \qquad (2.5)$$

Jadi jarak minimum (*minimum distance*) suatu kode linier dapat dipastikan tanpa harus menguji jarak antara semua kombinasi vektor pasangan kode. Kita hanya harus menguji bobot (*weight*) dari tiap vektor kode (tidak termasuk *all-zeros* vektor) di dalam *subspace*, *weight* yang minimum merupakan jarak yang minimum, d_{min}. Setara dengan hal itu, d_{min} merupakan nilai terkecil pada himpunan jarak antara vektor nol dengan masing-masing vektor sandi lainnya.

2.2.3. Koreksi dan pendeteksian kesalahan

Tugas dari pengawasandi, setelah menerima vektor r, adalah menaksir vektor kode yang dikirimkan U_i. Strategi yang optimal dari pengawasandi dapat dinyatakan dengan istilah algoritma *Maximum Likehood* sebagai berikut, diputuskan menuju ke U_i jika :

$$P(r|U_i) = \max_{\text{over all } U_j} P(r|U_j) \qquad (2.6)$$

Karena untuk saluran *Binary Symmetric Channel*(BSC), kemungkinan U_i berkenaan dengan r adalah berbanding terbalik dengan jarak antara r dan U_i, kita dapat tulis, diputuskan menuju ke U_i jika :

$$d(r, U_i) = \min_{\text{over all } U_j} d(r, U_j) \qquad (2.7)$$

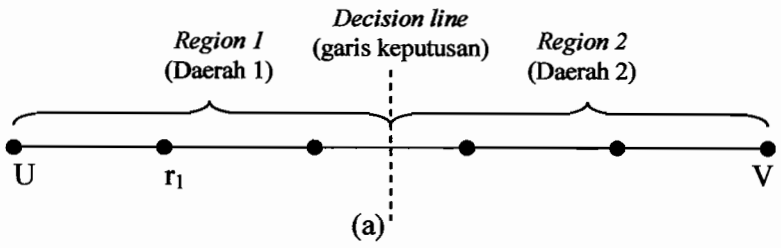
Dengan kata lain, pengawasandi menentukan jarak antara r dan masing-masing vektor kode yang dipancarkan yang mungkin U_i, dan memilih yang hampir sama dengan U_i, yaitu :

$$d(r, U_i) \leq d(r, U_j) \text{ untuk } i,j=1,.....,M \text{ dan } i \neq j \qquad (2.8)$$

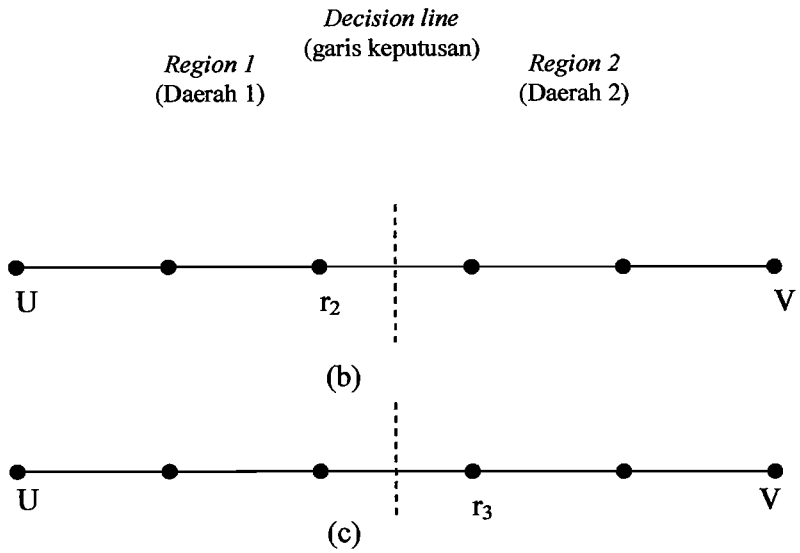
dengan $M = 2^k$ adalah ukuran dari himpunan vektor kode. Jika yang minimum tidaklah unik (yang minimum >1), maka dipilih salah satu secara acak.

Gambar 2.5 Jarak antara dua vektor kode U dan V ditunjukkan menggunakan suatu garis yang dikalibrasi di dalam jarak Hamming. Masing-masing titik hitam menandai suatu vektor kode yang terkena *error*. Gambar 2.5a menggambarkan vektor r_1, dengan jarak 1 dari U dan jarak 4 dari V. Pengawasandi pengoreksi *error*, mengikuti strategi maksimum *likelihood*, akan memilih U ketika menerima r_1. Jika r_1 adalah 1-bit vektor kode yang dipancarkan U dengan 1 bit *error*, maka pengawasandi sukses mengkoreksi kesalahan. Tetapi jika r_1 adalah vektor kode yang dipancarkan V dengan 4-bit *error*, maka pengawasandi salah memecahkan kode. Dengan cara yang sama, suatu kesalahan ganda di dalam transmisi U mungkin mengakibatkan vektor yang diterima r_2 adalah dengan jarak 2 dari U dan jarak 3 dari V, seperti ditunjukkan di dalam gambar 2.5b. Pengawasandi akan memilih U ketika menerima r_2. Kesalahan rangkap tiga di dalam transmisi U mengakibatkan suatu vektor diterima r_3 adalah dengan jarak 3 dari U dan jarak 2 dari V, seperti ditunjukkan di dalam gambar 2.5c. Pengawasandi akan memilih V ketika menerima r_3, sehingga salah dalam memecahkan kode.

Dari gambar 2.5 jelas bahwa jika tugas pengawasandi adalah mendeteksi kesalahan dan bukan koreksi, maka suatu vektor yang terkena *error*, dengan penanda titik hitam 1-bit, 2-bit, 3-bit, atau 4-bit, dapat dideteksi ketika vektor kode U benar-benar dipancarkan. Tetapi kesalahan dengan 5 *error* akan bersifat tidak bisa dideteksi karena dianggap sebagai vektor V sehingga dianggap tidak ada *error*.



Gambar 2.5 koreksi kesalahan dan pendeteksian kekuatan (a)vektor yang diterima r_1



Gambar 2.5 (lanjutan) koreksi kesalahan dan pendeteksian kekuatan
(b)vektor yang diterima r_2 (c)vektor yang diterima r_3

Dari gambar 2.5 kita dapat lihat bahwa kemampuan deteksi dan koreksi *error* (*error-detecting* dan *error-correcting*) suatu kode berhubungan dengan jarak minimum antara vektor kode. Garis keputusan di dalam gambar sama seperti proses dalam memecahkan kode. Dalam gambar 2.5 ukuran keputusan memilih U adalah jika r jatuh di dalam daerah 1, dan memilih V adalah jika r jatuh di dalam daerah 2. Jika $d_{min} = 5$, maka dapat mengkoreksi dua kesalahan. Secara umum, kemampuan *error-correcting*, t, suatu kode didefinisikan sebagai jumlah maksimum kesalahan yang dapat dikoreksi pada setiap *codeword*, dan ditulis :

$$t = \left\lfloor \frac{d\min - 1}{2} \right\rfloor \qquad (2.9)$$

dengan $\lfloor x \rfloor$ adalah integer terbesar yang $< x$

Suatu kode dapat digunakan untuk mendeteksi dan mengkoreksi kesalahan. Dari gambar 2.5 terlihat bahwa semua vektor yang diterima ditandai oleh suatu titik hitam (suatu vektor kode dirusak) dapat dikenali sebagai suatu

kesalahan. Oleh karena itu, *error-detecting capability* (kemampuan deteksi *error*), e, didefinisikan dalam kaitan dengan d_{min} sebagai berikut :

$$e = d_{min} - 1 \qquad (2.10)$$

Sebuah kode blok dengan jarak minimum d_{min} menjamin bahwa semua pola kesalahan d_{min}-1 atau lebih kecil kesalahan dapat dideteksi. Sebagai sebuah kode juga mampu mendeteksi pola kesalahan dengan d_{min} atau lebih banyak *error*. Pada kenyataannya, sebuah kode (n,k) mampu untuk mendeteksi pola kesalahan 2^n-2^k dengan panjang n. Ada sejumlah 2^n-1 memungkinkan pola-pola *error nonzero* dalam *space* 2^n n-tuples. Bahkan ada sedikit pola dari sebuah *codeword* yang *valid* dapat memperlihatkan satu pola kesalahan yang potensial. Ada pola kesalahan 2^k-1 yang sama seperti 2^k-1 *nonzero codewords*. Jika ada pola kesalahan 2^k-1 yang muncul, itu akan menyebabkan *codeword* U_i yang ditransmisikan menjadi *codeword* U_j. U_j tersebut akan diterima dan sindromnya adalah nol. Pengawasandi menerima U_j sebagai *codeword* yang ditransmisikan sehingga proses pengawasandi salah. Bagaimanapun juga ada 2^k-1 kesalahan tidak bisa dideteksi sebagai pola-pola *error*. Jika pola kesalahan tidak sama dengan salah satu dari 2^k *codewords*, suatu sindrom *test* pada vektor r yang diterima menghasilkan suatu *nonzero* sindrom, dan kesalahan dideteksi. Ada kalanya 2^n-2^k pola *error* yang pasti terdeteksi. Untuk n yang banyak $2^k << 2^n$, hanya ada sebagian kecil dari pola-pola *error* yang tidak terdeteksi.

Jarak Hamming sandi orthogonal 256 x 256 adalah sama untuk semua kata sandi yaitu : $\frac{1}{2}256$ atau sebanyak 128, sehingga jarak minimum sandi orthogonal yaitu : dmin = $\frac{1}{2}256$ atau dmin = 128. Kemampuan deteksi dan koreksi sandi orthogonal 8 bit, dengan menggunakan persamaan (2.9) dan persamaan (2.10), yaitu :

Nilai dmin adalah :

$$\text{dmin} = \frac{1}{2}256$$

$$dmin = 128$$

Sehingga kemampuan mengkoreksinya adalah :

$$t = \left\lfloor \frac{d\,min - 1}{2} \right\rfloor$$

$$t = \left\lfloor \frac{128 - 1}{2} \right\rfloor \equiv 63$$

Sedangkan kemampuan untuk mendeteksinya adalah :

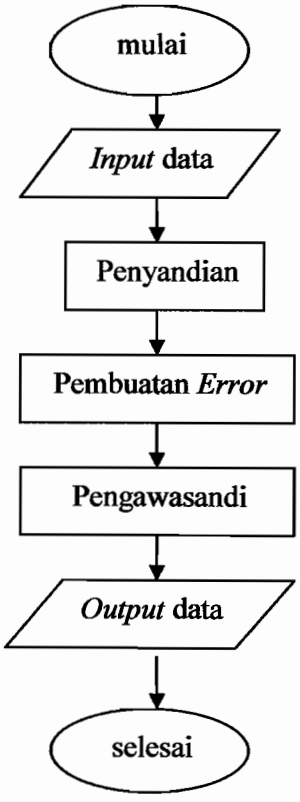
$$e = d_{min} - 1$$

$$e = 127$$

BAB III
PERANCANGAN SISTEM

3.1 Gambaran Umum Program

Program ini dibuat menggunakan MATLAB dengan diagram alir program utama seperti gambar 3.1. Program akan mendeteksi *error* dari *input* / masukan data berupa 8 bit dengan kata sandi sebanyak 256 bit. *User* atau pengguna dapat merubah kata sandi dengan mengubah satu bit atau lebih dalam kata sandi tersebut. Kemudian kata terima tersebut dideteksi, setelah dilakukan proses deteksi maka program akan mengeluarkan informasi "Ada *error*" atau "Tidak ada *Error*".



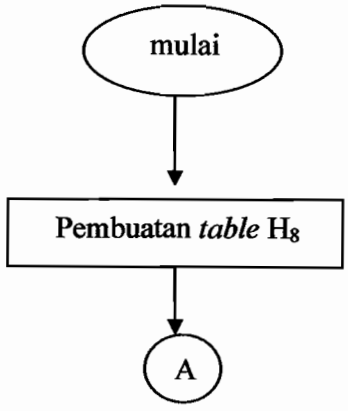
Gambar 3.1. Program utama

3.2 Penyandian

Dalam proses penyandian, data dimasukkan sebanyak 8 bit dan pada kata sandi akan menjadi 256 bit. Dalam permasalahan tersebut terlihat jelas *output* yang dikeluarkan dari data set adalah dihitung berdasarkan jumlah *input* data set. Secara umum, matriks katasandi (*Hadamard Matrix*), H_8 , dengan dimensi 2^8 x 2^8 untuk data *8*-bit dapat dibangun dari matriks H_7 , sebagai berikut :

$$H_8 = \begin{bmatrix} H_7 & H_7 \\ H_7 & \overline{H_7} \end{bmatrix}$$

Dengan menggunakan matriks H_8 tersebut, maka data 8-bit dapat diubah menjadi katasandi orthogonal yang masing-masing terdiri dari 256 bit. Setiap katasandi dalam matriks tersebut mempunyai digit yang sama sebanyak digit yang tidak sama. Adapun diagram alir untuk program penyandian ditunjukkan pada gambar 3.2 :



Gambar 3.2. Proses pembuatan *table* H_8

Dalam proses pembuatan *table* untuk membangkitkan H_8 diperlukan H_7. Untuk Membangkitkan H_7 diperlukan H_6. Untuk membangkitkan H_6 diperlukan H_5. Untuk membangkitkan H_5 diperlukan H_4. Untuk membangkitkan H_4

diperlukan H_3. Untuk membangkitkan H_3 diperlukan H_2. Untuk membangkitkan H_2 diperlukan H_1. Untuk perumusannya sebagai berikut :

$$H_1 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \tag{3.1}$$

$$H_2 = \begin{pmatrix} H_1 & H_1 \\ H_1 & \overline{H_1} \end{pmatrix} \tag{3.2}$$

$$H_3 = \begin{pmatrix} H_2 & H_2 \\ H_2 & \overline{H_2} \end{pmatrix} \tag{3.3}$$

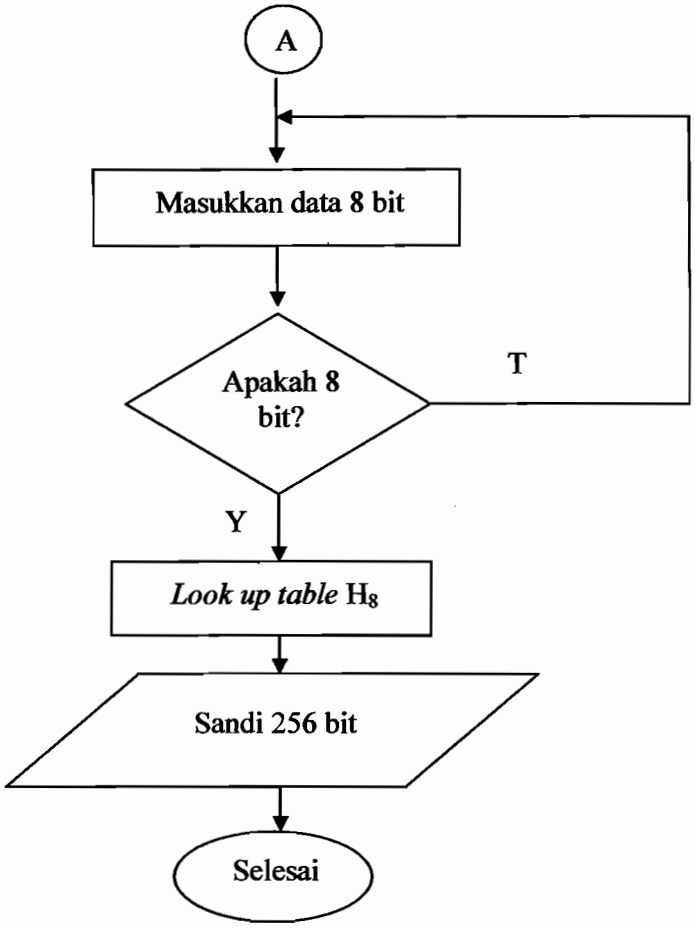
$$H_4 = \begin{pmatrix} H_3 & H_3 \\ H_3 & \overline{H_3} \end{pmatrix} \tag{3.4}$$

$$H_5 = \begin{pmatrix} H_4 & H_4 \\ H_4 & \overline{H_4} \end{pmatrix} \tag{3.5}$$

$$H_6 = \begin{pmatrix} H_5 & H_5 \\ H_5 & \overline{H_5} \end{pmatrix} \tag{3.6}$$

$$H_7 = \begin{pmatrix} H_6 & H_6 \\ H_6 & \overline{H_6} \end{pmatrix} \tag{3.7}$$

$$H_8 = \begin{pmatrix} H_7 & H_7 \\ H_7 & \overline{H_7} \end{pmatrix} \tag{3.8}$$

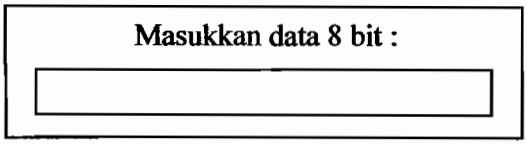


Gambar 3.3. Proses penyandian

Gambar 3.3 menjelaskan bahwa dengan *look up table* data 8 bit diubah menjadi katasandi orthogonal 256 bit.

Tampilan untuk proses penyandian :

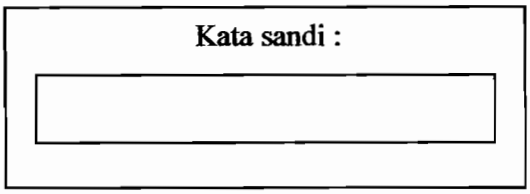
Masukkan data 8-bit =



Gambar 3.4. masukkan data 8 bit

Pada kotak di dalam gambar 3.4 berisi data sebanyak 8 bit, kemudian diubah dengan cara *look up table* menjadi :

Kata sandi =

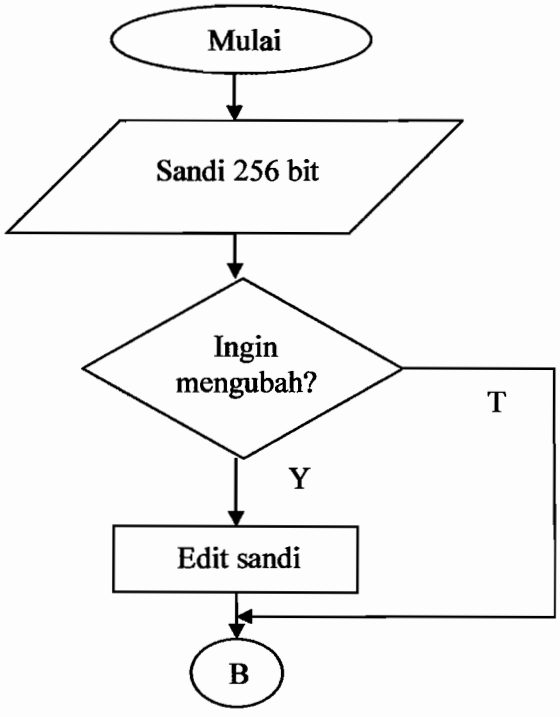


Gambar 3.5. kata sandi

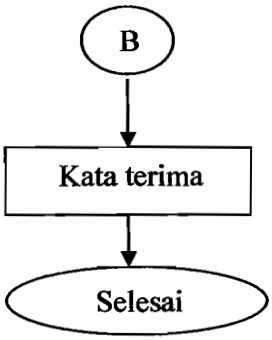
Pada kotak di dalam gambar 3.5 berisi katasandi sebanyak 256 bit. Kata sandi inilah yang akan dikirim melalui *channel*. Apabila pada *channel* terjadi *error*, kemudian mensimulasi adanya *error*. Dengan adanya *error* ini maka dibuat fasilitas pembuat *error*.

3.3 Pembuatan *error*

Dalam pembuatan *error* langkah-langkah yang akan dilakukan yaitu dengan cara langsung mengganti kata sandi, adapun diagram alir pembuatan *error* sebagai berikut :



Gambar 3.6 Pembuatan *error*

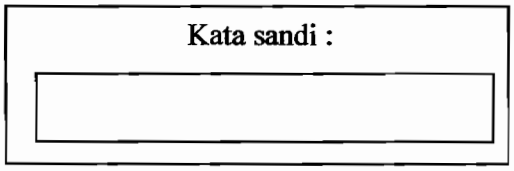


Gambar 3.6 (lanjutan) Pembuatan *error*

Untuk pembuatan *error* akan dibuat suatu fasilitas untuk merubah hasil dari kata sandi, maka pengguna dapat mengganti atau merubah angka 0 dan 1 pada posisi mana saja tergantung dari keinginan pengguna.

Tampilan untuk gambar 3.6. pembuatan *error* seperti di bawah ini :

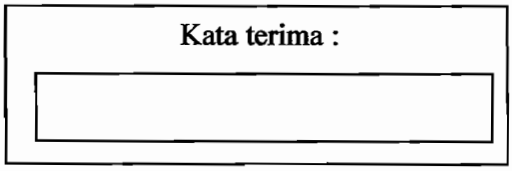
Kata sandi =



Gambar 3.7. kata sandi pada pembuatan *error*

Pada kotak di dalam gambar 3.7. berisi katasandi sebanyak 256 bit, kemudian diubah menjadi :

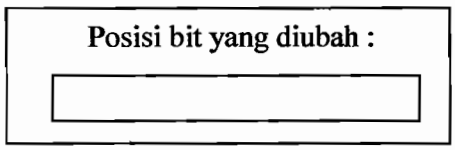
Kata terima =



Gambar 3.8. kata terima pada pembuatan *error*

Pada kotak di dalam gambar 3.8. berisi kata terima sebanyak 256 bit

Rekapitulasi posisi *error* =

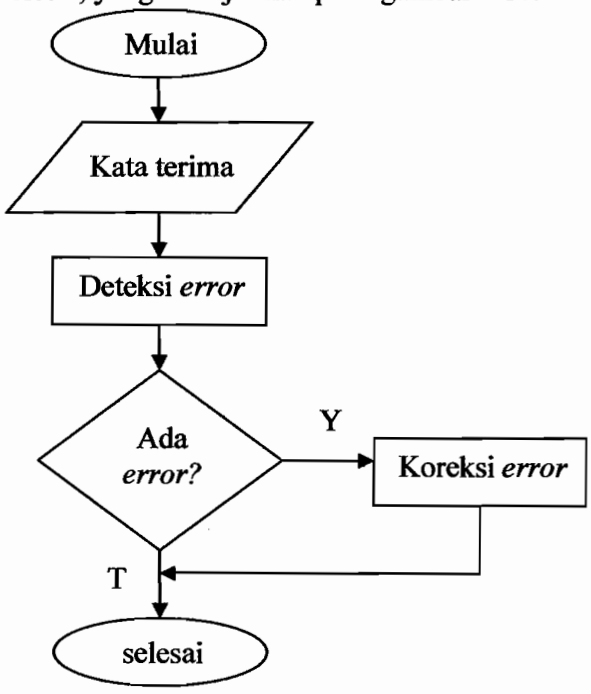


Gambar 3.9. Posisi bit yang diubah

Pada kotak di dalam gambar 3.9. berisi posisi bit mana saja yang telah diubah.

3.4. Pengawasandi

Pada proses pengawasandi kata terima akan dideteksi apa ada *error* atau tidak, jika ada *error* maka akan ada koreksi *error*. Jika dideteksi tidak ada *error* maka proses pengawasandi selesai, yang ditunjukkan pada gambar 3.10.

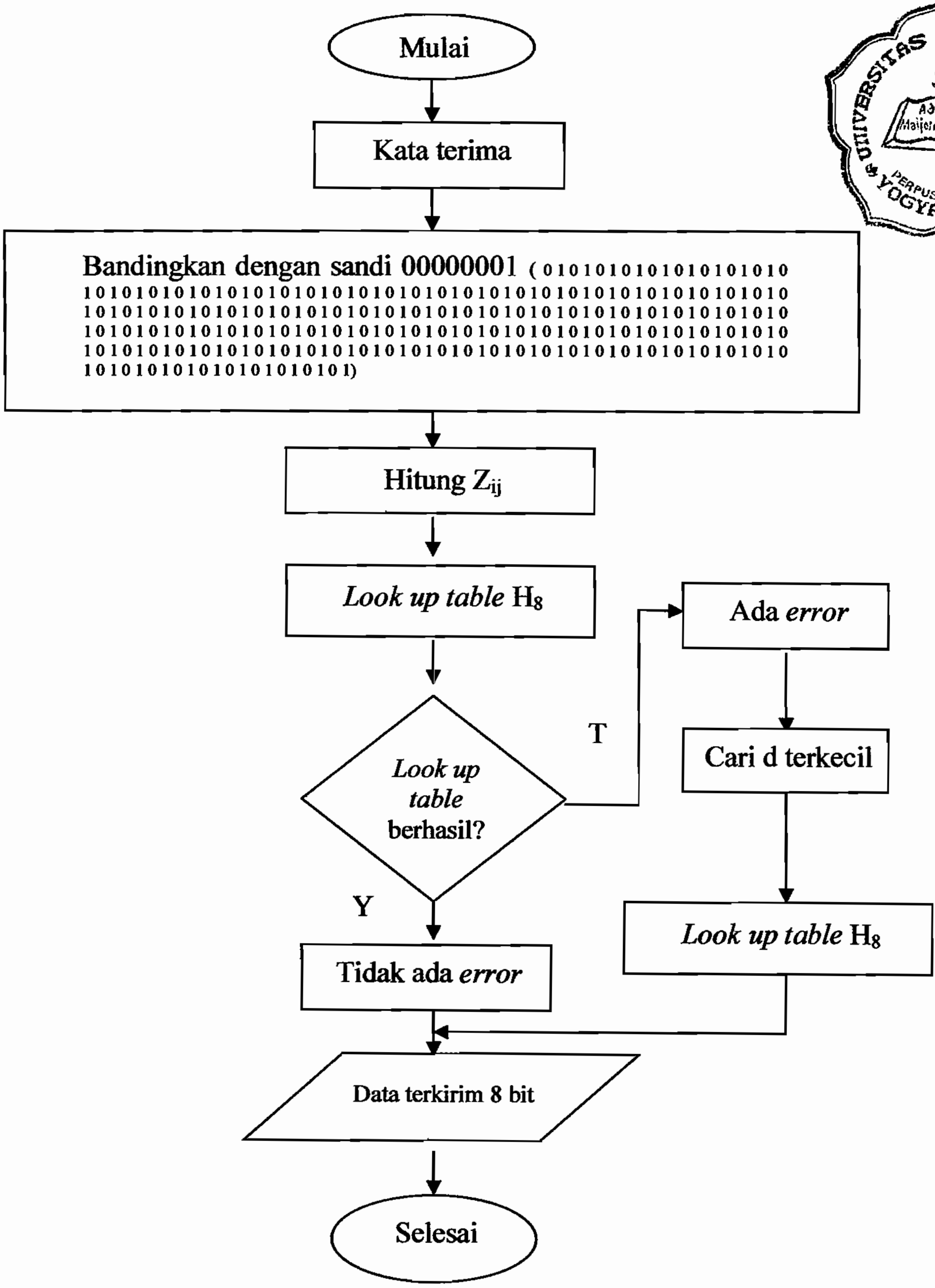


Gambar 3.10. Pengawasandi

3.5. Deteksi *error*

Pada proses deteksi *error*, kata terima dibandingkan dengan salah satu sandi dari *lookup table* untuk mendapatkan nilai Z_{ij}. Pada proses *lookup table*, jika

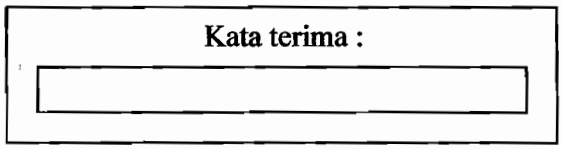
berhasil berarti tidak ada *error*. Tapi jika *lookup table* gagal berarti ada *error*. Apabila ada *error*, maka akan dicari nilai dmin (jarak minimum dari sandi), kemudian dilakukan proses *lookup table* yang akan menghasilkan data terkirim 8 bit. Adapun gambar untuk deteksi *error* itu ditunjukkan pada gambar 3.11. :



Gambar 3.11. Deteksi *error*

Tampilan untuk gambar 3.11. deteksi *error* seperti di bawah ini :

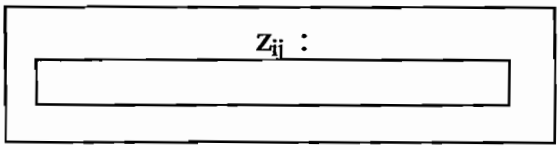
Kata terima =



Gambar 3.12. kata terima pada deteksi *error*

Pada kotak di dalam gambar 3.12. di atas berisi kata terima sebanyak 256 bit.

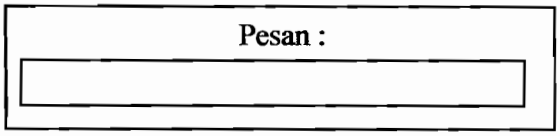
Nilai z_{ij} =



Gambar 3.13. Nilai Zij

Pada kotak di dalam gambar 3.13. di atas berisi nilai z_{ij}.

Pesan =

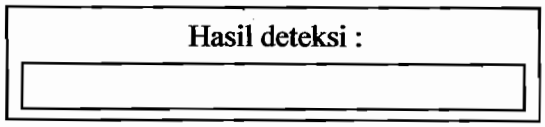


Gambar 3.14. Pesan

Pada kotak di dalam gambar 3.14. di atas berisi pesan "*Look up* berhasil" atau "*Look up* gagal".

Jika pesan "*Look up* berhasil" maka akan ada informasi sebagai berikut :

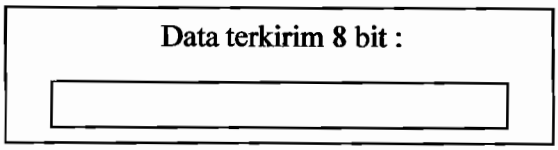
Hasil deteksi =



Gambar 3.15. Hasil deteksi *Look up* berhasil

Pada kotak di dalam gambar 3.15. di atas berisi "Tidak ada *error*".

Data terkirim 8 bit =

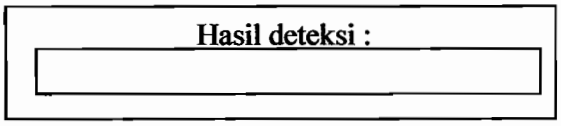


Gambar 3.16. Data terkirim 8 bit *Look up* berhasil

Pada kotak di dalam gambar 3.16. di atas berisi data 8 bit yang terkirim.

Jika berisi pesan "gagal *lookup table*" maka akan ada informasi sebagai berikut :

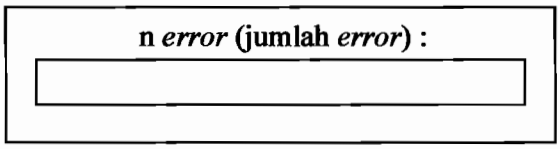
Hasil deteksi =



Gambar 3.17. Hasil deteksi gagal *lookup table*

Pada kotak di dalam gambar 3.17. di atas berisi "Ada *error*".

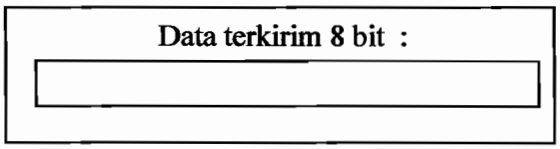
n *error* (jumlah *error*) =



Gambar 3.18. Jumlah *error*

Pada kotak di dalam gambar 3.18. di atas berisi jumlah perbedaan digit pada proses *lookup table*.

Data terkirim **8** bit =



Gambar 3.19. Data terkirim 8 bit gagal *lookup table*

Pada kotak di dalam gambar 3.19. di atas berisi data **8** bit yang terkirim.

BAB IV
ANALISA DAN PEMBAHASAN

4.1 Struktur Program

Sistem disusun dalam pengembangan program yang dibuat dengan menggunakan MATLAB. *File-file* yang berisi kode-kode program ditulis dengan menggunakan *m-editor* dan disimpan dalam *m-files*. Dalam sistem ini, perintah-perintah program hanya disimpan dalam *file-file* berikut ini :

a). file '*coding.m*', merupakan program utama

b). file '*BuatEror.fig*' dan '*BuatEror.m*' , digunakan sebagai interaksi bagi pemakai untuk membuat simulasi adanya *error* dengan cara melakukan perubahan langsung pada data yang dikirim atau yang diterima.

Secara keseluruhan program dikelompokkan dalam beberapa bagian yang menunjukkan proses yang dikerjakan, dengan rincian sebagai berikut :

1). Pembuatan matriks katasandi (*hadamard matrix*)

2). Proses memasukkan data (*entry data*)

3). Proses penyandian

4). Pembuatan *Error*

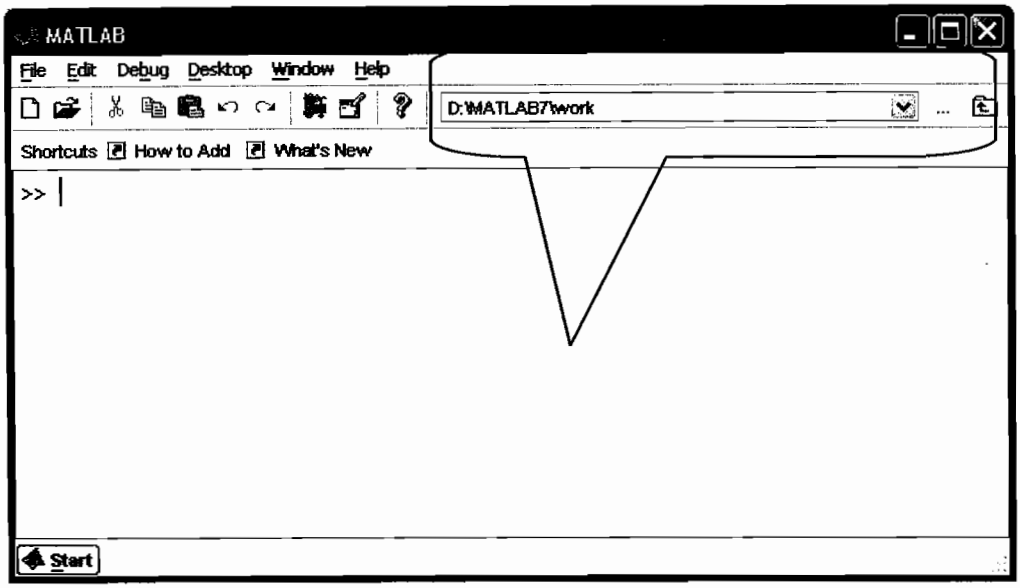
5). Deteksi *Error*

6). Proses pengawasandi

4.2 Menjalankan Program

Untuk menjalankan kode-kode program dalam *file coding.m*, *Matlab* harus dijalankan dulu, kemudian cari *folder* lokasi *file* tersebut disimpan. Selanjutnya program *coding.m* bisa dijalankan dengan mengetikkan perintah atau nama *file* program pada *command editor*, ditunjukkan pada gambar 4.1.:

```
>> coding <ENTER>
```



Gambar 4.1. Tampilan Awal Matlab

Tampilan awal sebagai *title* program adalah "CODEC ORTHOGONAL 8 BIT". Selanjutnya program akan meminta *user* untuk mengetikkan data 8 bit yang menjadi masukan program, yaitu data 8 bit yang akan dikirimkan ke lokasi tujuan.

4.3 Pembuatan matriks katasandi (*hadamard matrix*)

Matrik katasandi merupakan matrik berukuran 256x256, matrik ini digunakan untuk melakukan proses penyandian dari data yang dimasukkan. Matrik katasandi disusun dari matrik dengan ordo 2x2, dengan menggunakan persamaan (2.3).

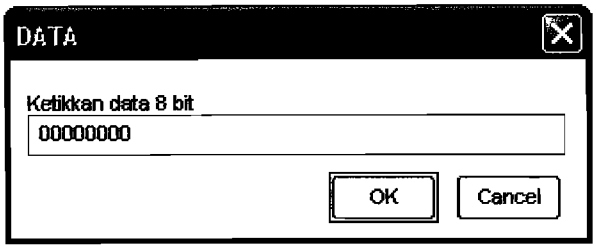
Diimplementasikan dalam perintah program sebagai berikut :

```
%% pembuatan tabel lookup

H1=[0 0; 0 1];
H2=[H1 H1; H1 ~H1];
H3=[H2 H2; H2 ~H2];
H4=[H3 H3; H3 ~H3];
H5=[H4 H4; H4 ~H4];
H6=[H5 H5; H5 ~H5];
H7=[H6 H6; H6 ~H6];
H8=[H7 H7; H7 ~H7];
```

Hasil akhir dari kode-kode program tersebut adalah didapatkan matrik katasandi H_8 dengan ukuran 256x256. Matrik ini yang digunakan untuk melakukan penyandian data yang dimasukkan sebelum data tersebut dikirimkan ke lokasi tujuan.

4.4 Proses memasukkan data (*entry data*)

Setelah program dijalankan, maka tampilan awal yang merupakan interaksi dengan *user* adalah meminta *user* untuk memasukkan data 8 bit yang menjadi masukan program, dalam tampilan seperti ditunjukkan gambar 4.2.



Gambar 4.2. Tampilan *entry data*

User diminta untuk mengetikkan data 8 bit dalam bentuk data biner, dengan hanya menggunakan simbol angka 1 (satu) dan simbol angka 0 (nol). *Default* data seperti yang tampak pada gambar 4.2. adalah "00000000".

Pada proses memasukkan data, terdapat 2 (dua) kriteria yang merupakan validasi data yang dimasukkan, yaitu :

1) Panjang data harus terdiri dari 8 bit. Kesalahan akan terjadi jika panjang data $\neq$ 8, dan bukan bit. Kesalahan menyebabkan program akan meminta *user* mengulangi kembali pengetikan data masukan.
2) Setiap digit harus terdiri dari simbol angka 1 (satu) atau simbol angka 0 (nol), kesalahan akan diberikan komentar *"ada kesalahan pada digit ke ..."*, dan selanjutnya akan diminta mengulangi mengetikan data masukan kembali.

Selengkapnya proses ini dituliskan dalam bagian program berikut ini :

```
%%masukkan data 8 bit, 0-255
adasalah='ya';
while adasalah=='ya'
    cdata=cellstr('00000000');
    cdata=inputdlg('Ketikkan data 8 bit','DATA',1,cdata);
    %%cek validasinya
    sdata=char(cdata);
    if length(sdata)==8
       %%cek digit biner 8 bit masukkan
       adasalah='no';
       for i1=1:8
            if (sdata(i1)~='0') && (sdata(i1)~='1')
               adasalah='ya';
               disp(['ada kesalahan pada digit ke...
                        ',num2str(i1)]);
               disp(' ');
               break;
            end;
       end;
    else
       %% jika panjang data tidak 8 bit
       disp(['Panjang data yang diketikkan ...
                  ',num2str(length(sdata)),' ..digit']);
       disp('Ketikkan data tepat 8 bit / 8 digit biner');
       disp(' ');
    end;
end;
```

Bagian akhir dari proses ini, program akan menampilkan data masukan dalam bentuk bilangan desimal. Hal ini dimaksudkan agar lebih mudah diamati oleh *user*. Bentuk tampilan dari program adalah sebagai berikut :

```
Data entry  : 00100100(BINER)
Data entry  : 36(DESIMAL)
```

Diimplementasikan dalam perintah program sebagai berikut

```
%% tampilkan entry data //DATA//
data=bin2dec(cdata);
disp(['Data entry  : ',char(cdata),'(BINER)']);
disp(['Data entry  : ',num2str(data),'(DESIMAL)']);
```

4.5 Proses penyandian

Selanjutnya terhadap data yang dimasukan akan dilakukan proses penyandian. Matrik katasandi yang berukuran 256x256 digunakan sebagai tabel *lookup* dari data yang dimasukkan.

Sebagai misal data masukan "00000001" (dalam bentuk desimal adalah bilangan 1 (satu)), maka proses penyandian dilakukan dengan cara mengambil data pada baris ke-1 (pertama), sehingga bilangan 1 akan dikodekan dengan :

"01010101010101010101010101010101010101010101010101010101010101010101
01010101010101010101010101010101010101010101010101010101010101010101
10101010101010101010101010101010101010101010101010101010101010101010
01010101010101010101010101010101010101010101010101010101010101010101"

Pernyataan programnya dituliskan sebagai berikut :

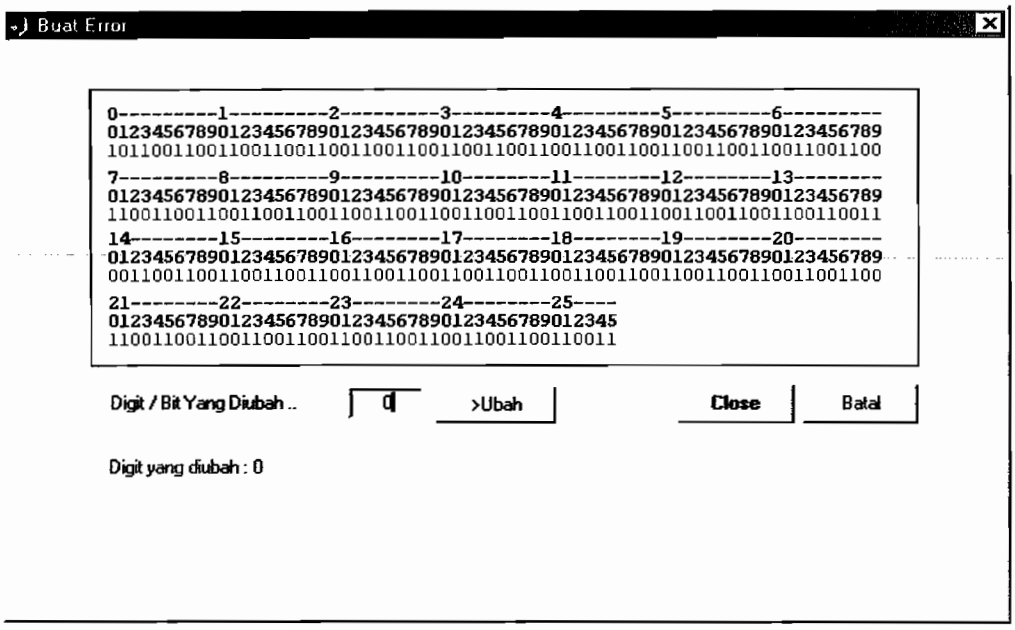
```
%%tentukan sandi 256 bit //SANDI//
sandi=H8(data+1,1:256);
```

Dituliskan data+1, dikarenakan data masukan akan terdiri dari bilangan desimal 0 (nol) hingga 255, sementara indek matrik katasandi dihitung mulai nomor 1 (satu) hingga 256.

4.6 Pembuatan *Error*

Pembuatan *error* dimaksudkan sebagai bentuk simulasi, kemungkinan terjadinya distorsi data selama proses pengiriman dari tempat asal ke tempat tujuan. Pada program ini terjadinya *error* dilakukan dengan melakukan penyuntingan (*"editing"*) data yang dikirimkan, yaitu hasil penyandian yang berukuran 256 bit.

Proses pembuatan *error* dilakukan dengan menjalankan program *"BuatError"* dengan mengirimkan variabel *cKirim* dan *cTerima* menggunakan media *WorkSpace MathLab.* Tampilan GUI untuk pembuatan *error* ditunjukkan pada gambar 4.3.



Gambar 4.3 Tampilan pembuatan error

Proses pembuatan *error* dengan cara mengetikkan Data /Bit ke berapa yang akan diubah, kemudian *Click* pada tombol ubah, maka secara otomatis bit yang bernilai 1 akan berubah menjadi 0, demikian juga sebaliknya. *Button "Batal"* digunakan untuk mengembalikan nilai *cTerima* sama dengan *cKirim*, jadi proses pembuatan *error* dibatalkan. Sedangkan jika *click* pada *button "close"* maka proses pembuatan *error* akan diakhiri dan diteruskan untuk dilakukan proses lebih lanjut.

Secara umum perintah yang dijalankan pada saat *click button "close"* adalah sebagai berikut :

```
      % --- Executes on button press in pushbutton1.
      function    pushbutton1_Callback(hObject,    eventdata,
handles)
      %% kembalikan terima ke workspace
      cterima = strcat(get(handles.text3,'string'),
      get(handles.text6,'string'),...
      get(handles.text9,'string'),get(handles.text12,'string
'))
      assignin('base','cterima',cterima);
      %% kembalikan proses
      handles.output = get(hObject,'String');      %%--Close-
-
      guidata(hObject, handles);
```

```
uiresume(handles.figure1);
```

Sedangkan proses validasinya dilakukan pada perintah sebagai berikut :

```
% --- Executes on key press over figure1 with
%no controls selected.
function figure1 KeyPressFcn(hObject, eventdata,
handles)
% Check for "enter" or "escape"
if isequal(get(hObject,'CurrentKey'),'escape')
    handles.output = 'Batal';
    guidata(hObject, handles);
    uiresume(handles.figure1);
end
if isequal(get(hObject,'CurrentKey'),'return')
    uiresume(handles.figure1);
end

function edit1 Callback(hObject, eventdata, handles)
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata,
handles)
salahInput = 't';
c1 = get(handles.edit1,'string');
n1 = length(c1);
if n1<1
    salahInput = 'y';
else
    for i1=1:n1
        if (c1(i1)<'0') || (c1(i1)>'9')
            salahInput = 'y';
        end;
    end;
    if salahInput == 't'
        d1 = str2num(c1);
        if (d1<0) || (d1>255)
            salahInput = 'y';
        end;
    end;
end
%%----------%% text 3 6 9 12
cterima = strcat(get(handles.text3,'string'),...
get(handles.text6,'string'),...
get(handles.text9,'string'),...
get(handles.text12,'string'));
if salahInput == 't'
    if cterima(d1+1)=='1'
        cterima(d1+1)='0';
    else
        cterima(d1+1)='1';
    end;
```

```
        set(handles.text3,'string',cterima(1:70));
        set(handles.text6,'string',cterima(71:140));
        set(handles.text9,'string',cterima(141:210));
        set(handles.text12,'string',cterima(211:256));
        %% cek digit diubah ---
        ckirim = char(evalin('base','ckirim'));
        adaError='no';
        digitUbah='';
        for i1=1:256
            if ckirim(i1) ~= cterima(i1)
                if adaError=='no'
                    digitUbah=['Digit yang diubah :
    ',num2str(i1-1)];
                    adaError='ya';
                else
                    digitUbah=[digitUbah,',   ',num2str(i1-
1)];
                end;
            end;
        end;
        set(handles.text18,'string',digitUbah);
    else
       errordlg('Ketikkan Nilai Bilangan antara 0 s.d.
    255','Salah Input','modal');
    end
```

Hasil pembuatan *error* ini, yang akan diteruskan pada tujuan, sehingga deteksi *error* akan dilakukan terhadap hasil pembuatan *error* ini.

4.7 Pengawasandi

4.7.1 Deteksi *Error*

Langkah awal yang dilakukan sebelum melakukan deteksi *error* data yang diterima, program akan menyusun tabel data yang digunakan untuk melakukan deteksi adanya *error* pada data yang diterima. Pada program yang dirancang, tabel data yang digunakan untuk deteksi *error* adalah data 00000001 dengan kata sandi sebagai berikut :

```
01010101010101010101010101010101010101010101010101010101010101010101010101010101
10101010101010101010101010101010101010101010101010101010101010101010101010101010
01010101010101010101010101010101010101010101010101010101010101010101010101010101
10101010101010101010101010101010101010101010101
```

Dengan cara membandingkan setiap digit data yang diterima dengan digit-digit pada dari tabel data yang digunakan untuk deteksi *error*, maka dapat dihitung besarnya nilai Z_{ij}. Menggunakan persamaan (2.2).

Perintah program untuk menentukan digit biner yang tidak bersesuaian :

```
%%cek error dari //TERIMA//
digitBenar=0;
digitSalah=0;
adaError='no';
digitUbah='';
for i1=1:256
    if K8(i1)==terima(i1)
        digitBenar=digitBenar+1;
    else
        digitSalah=digitSalah+1;
    end;
    if sandi(i1)~=terima(i1)
        if adaError=='no'
            digitUbah=['Digit yang diubah : ',...
num2str(i1-1)];
            adaError='ya';
        else
            digitUbah=[digitUbah,', ',num2str(i1-1)];
        end;
    end;
end;
```

Perintah program untuk menghitung nilai Z_{ij} dituliskan sebagai berikut :

```
%% tentukan error berdasarkan nilai Z
zij=abs((digitBenar-digitSalah)/256);
disp('Sandi Kata Terima :');
disp(baris1);
disp(baris2);
disp(char(cterima));   karakter biner
disp(' ');
disp(['Z ij              : ',num2str(zij)]);
disp(digitUbah);
disp(' ');
```

Kisaran nilai Z_{ij} tersebut bisa dikelompokkan sebagai berikut :

1) Nilai $Z_{ij} = 0$, jika digit yang sama berjumlah sama dengan digit yang tidak sama atau jumlah digit yang sama dan jumlah digit yang tidak sama berjumlah 128
2) Nilai $0 < Z_{ij} < 1$, hal ini terjadi jika ada digit yang berbeda antara data diterima dengan tabel pembanding, dan jumlah digit yang sama ≠ jumlah digit yang berbeda.
3) Nilai $Z_{ij} = 1$, jika semua digit dari data yang diterima sama dengan digit-digit dari tabel data untuk deteksi *error*

Nilai Z_{ij} digunakan untuk mengetahui sandi tersebut orthogonal atau tidak. Langkah berikutnya untuk membuat kesimpulan akhir data yang diterima, dilakukan dengan melakukan proses pengawasandi, untuk menentukan nilai data yang diterima dalam bentuk 8 bit. Jika berhasil *look up table* kesimpulannya tidak ada *error*, tetapi jika gagal *look up table* kesimpulannya ada *error*.

4.7.2 Proses Koreksi

Proses pengawasandi dilakukan apabila pada deteksi *error* didapatkan kesimpulan "tidak ada *error*" . Proses pengawasandi merupakan kebalikan dari proses penyandian, yaitu dilakukan dengan membandingkan data yang diterima dengan tabel *lookup* H_8 dan kemudian mengambil posisi indeks tabel sebagai nilai yang dikembalikan setelah proses pengawasandi.

Perintah programnya dituliskan sebagai berikut :

```
hasil = -1;
for i1=1:256
    if terima==H8(i1,1:256)
        %%hasil = index dari tabel H8
        hasil = i1-1;
        break;
    end;
end;
```

Pada pengujian proses pengawasandi pada nilai Z_{ij} didapatkan hasil sebagai berikut :

1). Proses *lookup* kembali pada tabel H_8 ditemukan hasil yang sesuai, yang selanjutnya dibuat kesimpulan "proses pengawasandi berhasil"

2). Proses *lookup* kembali pada tabel H_8 tidak ditemukan hasil yang sesuai, yang selanjutnya dibuat kesimpulan "proses pengawasandi gagal"

4.8 Hasil akhir data diterima (kataterima)

Penyajian data diterima oleh program ditentukan dari 2 (dua) proses sebelumnya, yaitu dengan cara menentukan nilai Z_{ij} dan proses pengawasandi sebagai kebalikan dari proses penyandian. Berdasarkan nilai Z_{ij} yang didapatkan maka diperoleh 2 (dua) kategori untuk membuat kesimpulan nilai akhir dari *kata terima* sebagai berikut :

1). Nilai $Z_{ij} = 0$ atau nilai $Z_{ij} = 1$

Pada nilai $Z_{ij} = 0$ atau nilai $Z_{ij} = 1$, dibuat kesimpulan sementara '*tidak ada error*', namun jika pada proses pengawasandi gagal diperoleh nilai data 8 bit yang sesuai, maka kesimpulan berikutnya dinyatakan sebagai *"proses lookup gagal"*, kemudian kata terima akan ditentukan dengan cara mencari dmin.

2). Nilai $0 < Z_{ij} < 1$

Pada ketegori ini, dipastikan terjadi *"error"*, setelah disimpulan ada *"error"* maka langkah berikutnya dilakukan dengan membandingkan data yang diterima dengan semua nilai pada tabel H8 (tabel *lookup* 256x256). Kata terima ditentukan berdasarkan nilai kesalahan digit yang terkecil, yang pada umumnya masih ada kesamaan dengan data yang dikirimkan.

Secara umum perintah program selengkapnya adalah sebagai berikut :

```
%%cek error dari //TERIMA//
digitBenar=0;
digitSalah=0;
adaError='no';
digitUbah='';
```

```
for i1=1:256
    if K8(i1)==terima(i1)
        digitBenar=digitBenar+1;
    else
        digitSalah=digitSalah+1;
    end;
    if sandi(i1)~=terima(i1)
        if adaError=='no'
            digitUbah=['Digit yang diubah : ',
                num2str(i1-1)];
            adaError='ya';
        else
            digitUbah=[digitUbah,', ',num2str(i1-1)];
        end;
    end;
end;

%% tentukan error berdasarkan nilai Z
zij=abs((digitBenar-digitSalah)/256);
disp('Sandi Kata Terima :');
disp(baris1);
disp(baris2);
disp(char(cterima));     %% kata //TERIMA// dalam bentuk
karakter biner
disp(' ');
disp(['Z ij              : ',num2str(zij)]);
disp(digitUbah);
disp(' ');

if (zij==0) | (zij==1)
    %% kembalikan terima (256 bit)
      %% menjadi (8 bit) menggunakan tabel H8
    adaError = 'no';
    hasil = -1;
    for i1=1:256
        if terima==H8(i1,1:256)
            %%hasil = index dari tabel H8
            hasil = i1-1;
            break;
        end;
    end;
    chasil=dec2bin(hasil);
    while length(chasil)<8
        chasil=strcat('0',chasil);
    end;

    disp(['Tidak ada error']);
    if hasil==(-1)
        disp('Proses Lookup Gagal');
        adaError='ya';
    else
        disp(['Kata Terima       ']);
        disp(['-----------       ']);
        disp(['Hasil (BINER)     : ',chasil]);
```

```
        disp(['Hasil (DESIMAL)    : ',num2str(hasil)]);
    end;
    disp(' ');
else
    adaError='ya';
end;

%% temukan katateriima jika adaerror dengan n salah
terkecil
if adaError=='ya'
    %% hitung d minimal pada H8 ----
    dmin = 300;
    hmin = 300;
    for i2=1:256
        digitBenar=0;
        digitSalah=0;
        K8 = H8(i2,1:256);
        for i1=1:256
            if K8(i1)==terima(i1)
                digitBenar=digitBenar+1;
            else
                digitSalah=digitSalah+1;
            end;
        end;
        if dmin > digitSalah
            dmin = digitSalah;
            hmin = i2-1;
        end;
        %%disp(strcat('error min :',...
           num2str(abs((digitBenar-digitSalah)/256))));
    end;
    %% tampilan biner ---
    chmin = dec2bin(hmin);
    while length(chmin)<8
        chmin = strcat('0',chmin);
    end;
    disp(['E min ( n error)   : ',num2str(dmin)]);
    disp(['Kata Terima        ']);
    disp(['-----------        ']);
    disp(['h min (BINER)      : ',chmin]);
    disp(['h min (DESIMAL)    : ',num2str(hmin)]);
end;
```

Tabel 4.1. Hasil uji coba dengan masukan data = 100 (desimal) = 01100100 (biner) dan *error* yang dikombinasi

Masukan 8 Bit (data dikirim) = 01100100 (biner) = 100 (desimal)

```
Kata Sandi Dikirim :

0---------1---------2---------3---------4---------5---------6---------7---------8---------9-----
01234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123450
00011110000111100001111000011111111000011110000111100001111000011110000111100001111000011110000

----10--------11--------12--------13--------14--------15--------16--------17--------18--------19
67890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012
00001111000011110000111100001111000011110000111100001111000011110000111100001111000011110000111100001111000

--------20--------21--------22--------23--------24--------25----
23456789012345678901234567890123456789012345678901234567890123456789012345
11110000111100001111000011110000000011110000111100001111000011110000111
```

No.	Jml *error*	Posisi bit yang diubah	Hasil hitungan				HASIL PROGRAM		KATA TERIMA (Dg Error Min)	
			Digit beda	Digit sama	Zij	dmin	Zij, pesan	dmin	Desimal	Biner
1	0	-	128	128	0	0	0, tidak ada *error*	0	100	01100100
2	1	0	129	127	0.0078125	1	0.0078125, ada *error*	1	100	01100100
3	1	254	127	129	0.0078125	1	0.0078125, ada *error*	1	100	01100100
4	2	0, 10	130	126	0.015625	2	0.015625, ada *error*	2	100	01100100
5	2	1, 255	128	128	0	2	0, ada *error*	2	100	01100100
6	3	0, 92, 158	129	127	0.0078125	3	0.0078125, ada *error*	3	100	01100100
7	3	46, 189, 235	129	127	0.0078125	3	0.0078125, ada *error*	3	100	01100100
8	4	0, 3, 90, 255	128	128	0	4	0, ada *error*	4	100	01100100
9	4	6, 100, 132, 200	124	132	0.03125	4	0.03125, ada *error*	4	100	01100100
10	10	10, 12, 25, 45, 54, 56, 58, 98, 152, 205	126	130	0.015625	10	0.015625, ada *error*	10	100	01100100
11	10	3, 30, 50, 59, 96, 108, 210, 211, 235, 254	132	124	0.03125	10	0.03125, ada *error*	10	100	01100100
12	25	1, 11, 13, 22, 23, 36, 60, 66, 74, 97, 106, 107, 116, 131, 139, 143, 147, 179, 185, 191, 196, 197, 199, 201, 229	125	131	0.0234375	25	0.023438, ada *error*	25	100	01100100
13	25	0, 2, 4, 15, 16, 18, 25, 26, 30, 35, 38, 95, 97, 98, 123, 125, 145, 154, 167, 179, 201, 220, 225, 229, 245	121	135	0.0546875	25	0.054688, ada *error*	25	100	01100100

Tabel 4.1. (lanjutan) Hasil uji coba dengan masukan data = 100 (desimal) = 01100100 (biner) dan *error* yang dikombinasi

No.	Jml *error*	Posisi bit yang diubah	Hasil hitungan				HASIL PROGRAM		KATA TERIMA (Dg Error Min)	
			Digit beda	Digit sama	Zij	dmin	Zij, pesan	dmin	Desimal	Biner
14	35	0, 1, 2, 5, 8, 9, 12, 15, 18, 25, 28, 32, 35, 45, 56, 67, 79, 95, 105, 109, 120, 125, 129, 131, 138, 145, 149, 156, 157, 168, 178, 189, 200, 208, 210	121	135	0.0546875	35	0.054688, ada *error*	35	100	01100100
15	35	0, 10, 12, 20, 23, 30, 34, 40, 45, 50, 60, 70, 80, 90, 100, 110, 120, 123, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230, 234, 240, 250, 253, 254, 255	133	123	0.0390625	35	0.039063, ada *error*	35	100	01100100
16	45	0, 5, 10, 15, 20, 22, 25, 30, 32, 35, 40, 42, 45, 50, 52, 55, 62, 65, 72, 75, 82, 85, 92, 95, 102, 105, 112, 115, 122, 125, 132, 135, 145, 155, 165, 175, 185, 195, 205, 210, 215, 225, 235, 245, 255	137	119	0.0703125	45	0.070313, ada *error*	45	100	01100100
17	54	0, 1, 3, 10, 11, 13, 21, 23, 31, 33, 41, 43, 51, 53, 61, 63, 71, 73, 81, 83, 91, 93, 101, 103, 111, 113, 121, 123, 131, 133, 141, 143, 151, 153, 161, 163, 171, 173, 181, 183, 191, 193, 201, 203, 211, 213, 221, 223, 231, 233, 241, 243, 251, 253	122	134	0.046875	54	0.046875, ada *error*	54	100	01100100
18	63	1, 3, 4, 6, 9, 11, 12, 14, 17, 19, 20, 22, 25, 27, 28, 30, 32, 34, 37, 39, 40, 42, 45, 47, 48, 50, 53, 55, 56, 58, 61, 63, 64, 66, 69, 71, 72, 74, 77, 79, 80, 82, 85, 87, 88, 90, 93, 95, 97, 99, 100, 102, 105, 107, 108, 110, 113, 115, 116, 118, 121, 123, 124	65	191	0.4921875	63	0.49219, ada *error*	63	100	01100100

Tabel 4.1. (lanjutan) Hasil uji coba dengan masukan data = 100 (desimal) = 01100100 (biner) dan *error* yang dikombinasi

No.	Jml *error*	Posisi bit yang diubah	Hasil hitungan				HASIL PROGRAM		KATA TERIMA (Dg Error Min)	
			Digit beda	Digit sama	Zij	dmin	Zij, pesan	dmin	Desimal	Biner
19	64	1, 3, 4, 6, 9, 11, 12, 14, 17, 19, 20, 22, 25, 27, 28, 30, 32, 34, 37, 39, 40, 42, 45, 47, 48, 50, 53, 55, 56, 58, 61, 63, 64, 66, 69, 71, 72, 74, 77, 79, 80, 82, 85, 87, 88, 90, 93, 95, 97, 99, 100, 102, 105, 107, 108, 110, 113, 115, 116, 118, 121, 123, 124, 126	64	192	0.5	64	0.5, ada *error*	64	1	00000001
20	66	0, 10, 11, 12, 20, 21, 22, 30, 31, 32, 40, 41, 42, 50, 51, 52, 60, 61, 62, 70, 71, 72, 80, 81, 82, 90, 91, 92, 100, 101, 102, 110, 111, 112, 120, 121, 122, 130, 131, 132, 140, 141, 142, 150, 151, 152, 160, 161, 162, 170, 171, 172, 180, 181, 182, 190, 191, 192, 200, 201, 202, 210, 211, 212, 220, 221	126	130	0.015625	66	0.015625, ada *error*	66	100	01100100
21	68	0, 4, 6, 7, 14, 16, 17, 24, 26, 27, 34, 36, 37, 44, 46, 47, 54, 56, 57, 64, 66, 67, 74, 76, 77, 84, 86, 87, 94, 96, 97, 104, 106, 107, 114, 116, 117, 124, 126, 127, 134, 136, 137, 144, 146, 147, 154, 156, 157, 164, 166, 174, 176, 184, 186, 194, 196, 204, 206, 214, 216, 224, 226, 234, 236, 244, 246, 254	132	124	0.03125	68	0.03125, ada *error*	68	100	01100100

Keterangan :
Digit beda adalah jumlah digit yang berbeda antara kata terima dengan acuan.
Digit sama adalah jumlah digit yang sama antara kata terima dengan acuan.
dmin adalah jarak minimum.

Dari tabel 4.1. hasil uji coba dengan masukan = 100 (desimal) = 01100100 (biner) dan *error* yang dikombinasi didapat :

1. Sandi kata terima hasil uji coba = hasil teori, berarti proses penyandian benar

2.a Zij = 0 dan pesan pada program mendeteksi bahwa "tidak ada *error*" ditunjukkan dalam tabel 1 yaitu hasil no. 1.

b.Zij = 0 dan pesan pada program mendeteksi bahwa "ada *error*" ditunjukkan dalam tabel 1 seperti hasil no. 5 dan no. 8.

c Pada tabel 1. kolom Zij, pesan menunjukkan bahwa semua *error* dapat dideteksi, hal ini sesuai dengan teori bahwa *error* $\leq$ 127 pasti dapat dideteksi.

3. Tabel 1 Menunjukkan bahwa semua *error* dapat dideteksi tapi ternyata tidak semua dapat dikoreksi seperti hasil no.19.

Tabel 4.2. Hasil uji coba dengan masukan data = 1 (desimal) = 00000001 (biner) dan *error* yang dikombinasi

Masukan 8 Bit (data dikirim) = 00000001 (biner) = 1 (desimal)

```
Kata Sandi Dikirim :

0---------1---------2---------3---------4---------5---------6---------7---------8---------9------
012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456
010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

---10--------11--------12--------13--------14--------15--------16--------17--------18--------19--
789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123
101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101

------20--------21--------22--------23--------24--------25----
456789012345678901234567890123456789012345678901234567890123456789012345
010101010101010101010101010101010101010101010101010101010101010101010101
```

No.	Jml *error*	Posisi bit yang diubah	Hasil hitungan				HASIL PROGRAM		KATA TERIMA (Dg Error Min)	
			Digit beda	Digit sama	Zij	dmin	Zij, pesan	dmin	Desimal	Biner
1	0	-	0	256	1	0	1, tidak ada *error*	0	1	00000001
2	1	2	1	255	0.9921875	1	0.99219, ada *error*	1	1	00000001
3	2	3, 11	2	254	0.984375	2	0.98438, ada *error*	2	1	00000001
4	3	23, 29, 58	3	253	0.9765625	3	0.97656, ada *error*	3	1	00000001
5	12	1, 14, 25, 28, 31, 52, 65, 89, 189, 214, 215, 254	12	244	0.90625	12	0.90625, ada *error*	12	1	00000001
6	28	0, 7, 9, 21, 28, 46, 72, 75, 85, 88, 94, 96, 97, 99, 100, 101, 105, 120, 123, 135, 154, 157, 161, 168, 208, 210, 223, 234	28	228	0.78125	28	0.78125, ada *error*	28	1	00000001
7	43	0, 1, 3, 5, 7, 8, 9, 10, 12, 21, 36, 39, 43, 45, 56, 69, 70, 73, 77, 79, 84, 89, 98, 99, 102, 111, 122, 123, 132, 144, 145, 150, 158, 159, 167, 169, 188, 189, 210, 222, 231, 234, 254	43	213	0.6640625	43	0.66406, ada *error*	43	1	00000001

Tabel 4.2. (lanjutan) Hasil uji coba dengan masukan data = 1 (desimal) = 00000001 (biner) dan *error* yang dikombinasi

No.	Jml *error*	Posisi bit yang diubah	Hasil hitungan				HASIL PROGRAM		KATA TERIMA (Dg Error Min)	
			Digit beda	Digit sama	Zij	dmin	Zij, pesan	dmin	Desimal	Biner
8	63	0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124	63	193	0.5078125	63	0.50781, ada *error*	63	1	00000001
9	64	1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99, 101, 103, 105, 107, 109, 111, 113, 115, 117, 119, 121, 123, 125, 127	64	192	0.5	64	0.5, ada *error*	64	0	00000000
10	84	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 17, 18, 19, 20, 21, 22, 23, 25, 28, 29, 30, 31, 32, 33, 34, 35, 36, 38, 39, 40, 41, 42, 44, 45, 47, 48, 50, 52, 53, 55, 56, 57, 58, 59, 61, 62, 63, 64, 66, 67, 68, 69, 71, 72, 73, 75, 76, 78, 80, 81, 82, 83, 84, 85, 86, 88, 89, 90, 91, 92, 93, 94, 96, 98, 99, 100, 101, 102, 104, 105	84	172	0.34375	84	0.34375, ada *error*	84	1	00000001
11	104	1, 3, 5, 7, 9, 11, 13, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 77, 79, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100, 102, 104, 106, 108, 110, 112, 114, 115, 117, 119, 121, 123, 125, 127, 129, 131, 133, 135, 137, 139, 140, 142, 144, 146, 148, 150, 220, 222, 224, 226, 227, 228, 229, 230, 233, 234, 236, 237, 239, 241, 242, 243, 244, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255	104	152	0.1875	60	0.1875, ada *error*	60	48	00110000

Tabel 4.2. (lanjutan) Hasil uji coba dengan masukan data = 1 (desimal) = 00000001 (biner) dan *error* yang dikombinasi

No.	Jml *error*	Posisi bit yang diubah	Hasil hitungan				HASIL PROGRAM		KATA TERIMA (Dg Error Min)	
			Digit beda	Digit sama	Zij	dmin	Zij, pesan	dmin	Desimal	Biner
12	126	1, 2, 4, 7, 9, 10, 12, 15, 17, 18, 20, 23, 25, 26, 28, 31, 33, 34, 36, 39, 41, 42, 44, 47, 49, 50, 52, 55, 57, 58, 60, 63, 65, 66, 68, 71, 73, 74, 76, 79, 81, 82, 84, 87, 89, 90, 92, 95, 97, 98, 100, 103, 105, 106, 108, 111, 113, 114, 116, 119, 121, 122, 124, 127, 129, 130, 132, 135, 137, 138, 140, 143, 145, 146, 148, 151, 153, 154, 156, 159, 161, 162, 164, 167, 169, 170, 172, 175, 177, 178, 180, 183, 185, 186, 188, 191, 193, 194, 196, 199, 201, 202, 204, 207, 209, 210, 212, 215, 217, 218, 220, 223, 225, 226, 228, 231, 233, 234, 236, 239, 241, 242, 244, 247, 249, 250	126	130	0.015625	2	0.015625, ada *error*	2	6	00000110
13	128	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223,	128	128	0	82	0, ada *error*	82	65	01000001

Tabel 4.2. (lanjutan) Hasil uji coba dengan masukan data = 1 (desimal) = 00000001 (biner) dan *error* yang dikombinasi

No.	Jml *error*	Posisi bit yang diubah	Hasil hitungan				HASIL PROGRAM		KATA TERIMA (Dg Error Min)	
			Digit beda	Digit sama	Zij	dmin	Zij, pesan	dmin	Desimal	Biner
13	128	224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255	128	128	0	82	0, ada *error*	82	65	01000001
14	135	0, 1, 2, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 65, 67, 69, 71, 73, 74, 75, 77, 78, 79, 81, 83, 85, 87, 89, 91, 93, 95, 98, 100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 129, 131, 133, 135, 137, 139, 141, 143, 145, 147, 149, 151, 153, 155, 157, 159, 162, 164, 166, 168, 170, 172, 174, 176, 178, 180, 182, 184, 186, 188, 190, 192, 193, 195, 197, 199, 201, 203, 205, 207, 209, 211, 213, 215, 217, 219, 221, 223, 226, 228, 230, 232, 234, 236, 238, 240, 242, 244, 245, 246, 248, 249, 250, 252, 254, 255	135	121	0.0546875	13	0.054688, ada *error*	13	32	00100000

Keterangan :
Digit beda adalah jumlah digit yang berbeda antara kata terima dengan acuan.
Digit sama adalah jumlah digit yang sama antara kata terima dengan acuan.
dmin adalah jarak minimum.

Dari tabel 4.2. hasil uji coba dengan masukan = 1 (desimal) = 00000001 (biner) dan *error* yang dikombinasi didapat :

1. Sandi kata terima hasil uji coba = hasil teori, berarti proses penyandian benar

2.a Zij = 1 dan pesan pada program mendeteksi bahwa "tidak ada *error*" ditunjukkan dalam tabel 2 yaitu hasil no. 1.

b. Pada kolom Zij, pesan menunjukkan bahwa semua *error* dapat dideteksi, hal ini sesuai dengan teori bahwa *error* $\leq$ 127 pasti dapat dideteksi.

3. Tabel 2 Menunjukkan bahwa semua *error* dapat dideteksi tapi ternyata tidak semua dapat dikoreksi seperti hasil no.9, no.11, no.12, no.13 dan no.14.

Tabel 4.3. Hasil uji coba dengan masukan data = 30 (desimal) = 00011110 (biner) dan *error* yang dikombinasi

Masukan 8 Bit (data dikirim) = 00011110 (biner) = 30 (desimal)

```
Kata Sandi Dikirim :

0---------1---------2---------3---------4---------5---------6---------7---------8---------9-----
012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345
001111001100001111000011001111000011110011000011110000110011110000111100110000111100001100111100

----10--------11--------12--------13--------14--------15--------16--------17--------18--------19
678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901
001111001100001111000011001111000011110011000011110000110011110000111100110000111100001100111100

--------20--------21--------22--------23--------24--------25----
234567890123456789012345678901234567890123456789012345678901234
001111001100001111000011001111000011110011000011110000110011110
```

No.	Jml *error*	Posisi bit yang diubah	Hasil hitungan				HASIL PROGRAM		KATA TERIMA (Dg Error Min)	
			Digit beda	Digit sama	Zij	dmin	Zij, pesan	dmin	Desimal	Biner
1	0	-	128	128	0	0	0, tidak ada *error*	0	30	00011110
2	1	85	127	129	0.0078125	1	0.0078125, ada *error*	1	30	00011110
3	2	79, 101	130	126	0.015625	2	0.015625, ada *error*	2	30	00011110
4	3	29, 42, 86	129	127	0.0078125	3	0.0078125, ada *error*	3	30	00011110
5	12	4, 6, 9, 12, 88, 101, 125, 144, 145, 168, 213, 245	130	126	0.015625	12	0.015625, ada *error*	12	30	00011110
6	25	2, 4, 16, 17, 23, 26, 29, 33, 40, 54, 64, 74, 79, 84, 94, 106, 112, 115, 120, 152, 183, 191, 209, 220, 228	127	129	0.0078125	25	0.0078125, ada *error*	25	30	00011110
7	44	0, 1, 4, 6, 14, 23, 30, 36, 39, 40, 43, 44, 50, 64, 65, 67, 77, 81, 89, 94, 95, 100, 103, 107, 112, 113, 128, 134, 135, 139, 142, 150, 160, 162, 172, 175, 176, 191, 203, 209, 212, 217, 242, 248	124	132	0.03125	44	0.03125, ada *error*	44	30	00011110
8	63	6, 7, 14, 20, 23, 24, 26, 28, 30, 40, 44, 46, 47, 50, 58, 66, 72, 75, 83, 86, 89, 95, 96, 100, 114, 118, 119, 125, 138, 140, 148, 154, 155, 156, 167, 173, 179, 182, 186, 193, 194, 195, 196, 200, 201, 202, 205, 211, 217, 218, 221, 222, 226, 229, 230, 235, 238, 239, 241, 246, 250, 251, 253	119	137	0.0703125	63	0.070313, ada *error*	63	30	00011110

Tabel 4.3. (lanjutan) Hasil uji coba dengan masukan data = 30 (desimal) = 00011110 (biner) dan *error* yang dikombinasi

No.	Jml *error*	Posisi bit yang diubah	Hasil hitungan				HASIL PROGRAM		KATA TERIMA (Dg Error Min)	
			Digit beda	Digit sama	Zij	dmin	Zij, pesan	dmin	Desimal	Biner
9	64	3, 7, 11, 18, 20, 30, 34, 43, 49, 50, 51, 52, 53, 59, 68, 69, 74, 80, 83, 89, 91, 92, 94, 97, 98, 100, 101, 108, 116, 118, 120, 122, 125, 128, 130, 137, 143, 148, 149, 155, 158, 167, 168, 170, 174, 178, 179, 180, 181, 183, 185, 187, 189, 197, 204, 212, 218, 224, 226, 228, 232, 235, 244, 249	124	132	0.03125	64	0.03125, ada *error*	64	0	00000000
10	85	1, 14, 15, 25, 26, 30, 33, 34, 36, 37, 38, 39, 41, 43, 44, 45, 48, 53, 54, 58, 59, 60, 62, 64, 66, 69, 71, 73, 75, 77, 79, 80, 81, 84, 85, 87, 93, 97, 99, 101, 102, 107, 110, 111, 113, 114, 117, 124, 130, 142, 150, 153, 154, 157, 158, 161, 163, 166, 169, 170, 171, 178, 181, 184, 192, 194, 196, 199, 200, 201, 206, 208, 225, 228, 232, 235, 239, 241, 242, 243, 246, 248, 249, 250, 251	117	139	0.0859375	85	0.085938, ada *error*	85	30	00011110
11	110	0, 2, 3, 4, 5, 6, 7, 12, 13, 17, 18, 19, 22, 24, 26, 28, 29, 32, 36, 39, 41, 42, 43, 45, 47, 51, 53, 58, 59, 61, 66, 69, 70, 71, 73, 78, 82, 83, 85, 86, 89, 93, 96, 98, 99, 102, 106, 108, 110, 111, 113, 119, 120, 123, 125, 126, 127, 128, 130, 131, 136, 138, 141, 142, 143, 148, 149, 150, 151, 155, 156, 159, 162, 163,	130	126	0.015625	110	0.015625, ada *error*	110	30	00011110

Tabel 4.3. (lanjutan) Hasil uji coba dengan masukan data = 30 (desimal) = 00011110 (biner) dan *error* yang dikombinasi

No.	Jml *error*	Posisi bit yang diubah	Hasil hitungan				HASIL PROGRAM		KATA TERIMA (Dg Error Min)	
			Digit beda	Digit sama	Zij	dmin	Zij, pesan	dmin	Desimal	Biner
11	110	167, 168, 170, 173, 176, 180, 182, 184, 187, 188, 189, 191, 192, 195, 196, 198, 199, 200, 203, 205, 216, 218, 222, 226, 228, 231, 235, 238, 239, 241, 245, 247, 250, 251, 253, 254	130	126	0.015625	110	0.015625, ada *error*	110	30	00011110
12	128	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127	128	128	0	128	0, ada *error*	128	0	00000000

Keterangan :
Digit beda adalah jumlah digit yang berbeda antara kata terima dengan acuan.
Digit sama adalah jumlah digit yang sama antara kata terima dengan acuan.
dmin adalah jarak minimum.

Dari tabel 4.3. hasil uji coba dengan masukan = 30 (desimal) = 00011110 (biner) dan *error* yang dikombinasi didapat :

1. Sandi kata terima hasil uji coba = hasil teori, berarti proses penyandian benar.

2.a Zij = 0 dan pesan pada program mendeteksi bahwa "tidak ada *error*" ditunjukkan dalam tabel 3 yaitu hasil no.1.

b. Zij = 0 dan pesan pada program mendeteksi bahwa "ada *error*" ditunjukkan dalam tabel 1 seperti hasil no.12.

c. Pada tabel 3 kolom Zij, pesan menunjukkan bahwa semua *error* dapat dideteksi, hal ini sesuai dengan teori bahwa *error* ≤ 127 pasti dapat dideteksi.

3. Tabel 3 Menunjukkan bahwa semua *error* dapat dideteksi tapi ternyata tidak semua dapat dikoreksi seperti hasil no.9 dan no.12.

BAB V
PENUTUP

5.1 Kesimpulan

Dari hasil simulasi yang dibuat dengan menggunakan program bantu matlab dan hasil perhitungan maka dapat disimpulkan sebagai berikut :

1) Program mampu menyandikan data 8 bit menjadi kata sandi 256 bit dengan benar.
2) Program mampu mendeteksi ada atau tidaknya *error* jika jumlah *error* ≤ 127.
3) Program mampu mengkoreksi *error* jika jumlah *error* ≤ 63.

5.2 Saran

Berdasarkan kesimpulan di atas, maka sangat dimungkinkan adanya penelitian atau pengujian lain yang bertujuan agar transmisi data bisa dilakukan dengan sebaik-baiknya. Beberapa hal yang menjadi saran penulis, sebagai berikut :

1) Ukuran data hasil penyandian relatif besar dibandingkan dengan data aslinya, sehingga perlu dipertimbangkan sistem penyandian yang lebih ramping.
2) Program bantu bisa dikembangkan lebih lanjut dengan pemrograman visual, agar mudah digunakan dan interaksinya dengan pemakai lebih mudah.

DAFTAR PUSTAKA

Agung P., Wahyu *Tips dan Trik MATLAB*, Penerbit ANDI Yogyakarta.2004

Arhami, Muhammad, S.Si., M.Kom dan Desiani, Anita, S.Si, M.Kom. *Pemrograman MATLAB*. Penerbit ANDI Yogyakarta.2004

Astrid, P. *Reduction of process simulation models: a proper orthogonal decomposition approach.* PhD dissertation, Eindhoven University of Technology, Department of Electrical Engineering, November 2004.

Astrid, P., S. Weiland, K.E.Willcox, and A.C.P.M. Backx. *Missing point estimation in models described by proper orthogonal decomposition.* In Proceedings of the 43rd IEEE Conference on Decision and Control, Paradise Island, Bahama, December 2004

Fasnet, *Modul Pelatihan MATLAB*. Pusat Pelatihan Teknologi Informasi. Fakultas Teknik Universitas Gadjah Mada. Yogyakarta. 2006

Lin, Shu dan J. Costello, Daniel, JR. *Error control coding Fundamentals and Applications*. Prentice-Hall, Inc. Englewood Cliffs, New Jersey.

Messner, William and Dawn Tilbury. *Control Tutorials for MatLab and Simulink. A Web Based Approach.* Addisson Wesley, Inc. 1999.

Ogata, Katsuhiko. *Modern Control Engineering. 3rd ed.* Prentice Hall International. 1997

Ogata, Katsuhiko. *Solving Control Engineering Problems with MatLab*. Englewood Cliffs, New Jersey : Prentice Hall Inc. 1994

Proakis, John G., Salehi, Masoud., Bauch, Gerhard., *Contemporary Communication Systems Using MATLAB and Simulink second edition*, Thomson Learning,Inc. Canada 2004.

Sklar, Bernard. *Digital Communications Fundamentals and Applications*.Prentice Hall, Upper Saddle River, New Jersey.1988

LAMPIRAN

LAMPIRAN
Program Inti

```
%%awal blok ulangi
ulangiTerus='Y';
while (ulangiTerus=='Y') | (ulangiTerus=='y')

clear;
clc;
disp('================================');
disp('---- CODEC ORTHOGONAL 8 BIT ----');
disp('--- KARYA  : CAHYO INDRATOMO ---');
disp('------- NIM  : 015114075 -------');
disp('================================');
disp(' ');

%% pembuatan tabel lookup
H1=[0 0; 0 1];
H2=[H1 H1; H1 ~H1];
H3=[H2 H2; H2 ~H2];
H4=[H3 H3; H3 ~H3];
H5=[H4 H4; H4 ~H4];
H6=[H5 H5; H5 ~H5];
H7=[H6 H6; H6 ~H6];
H8=[H7 H7; H7 ~H7];

%%masukkan data 8 bit, 0-255
adasalah='ya';
while adasalah=='ya'
    cdata=cellstr('00000000');
    cdata=inputdlg('Ketikkan data 8 bit','DATA',1,cdata);
    %%cek validasinya
    sdata=char(cdata);
    if length(sdata)==8
       %%cek digit biner 8 bit masukkan
       adasalah='no';
       for i1=1:8
            if (sdata(i1)~='0') && (sdata(i1)~='1')
               adasalah='ya';
               disp(['ada kesalahan pada digit ke ',num2str(i1)]);
               disp(' ');
               break;
            end;
       end;
    else
       %% jika panjang data tidak 8 bit
       disp(['Panjang data yang diketikkan
',num2str(length(sdata)),' ..digit']);
       disp('Ketikkan data tepat 8 bit / 8 digit biner');
       disp(' ');
    end;
end;

%% tampilkan entry data //DATA//
data=bin2dec(cdata);
disp(['Data entry  : ',char(cdata),'(BINER)']);
```

```
disp(['Data entry  : ',num2str(data),'(DESIMAL)']);
disp('');
%%tentukan sandi 256 bit //SANDI//
sandi=H8(data+1,1:256);
%% tampilkan data hasil encoding yang dikirim
baris1='';
baris2='';
tampil='';
for i1=1:256
    if mod(i1-2,10)==0
        baris1=[baris1,num2str(fix((i1-1)/10))];
    elseif length(baris1)<length(baris2)
        baris1=[baris1,'-'];
    end;
    baris2=[baris2,num2str(mod(i1-1,10))];
    tampil=[tampil,num2str(sandi(i1))];
end;
disp(' ');
disp('Sandi Kata Dikirim : ');
disp(baris1);
disp(baris2);
disp(tampil);
disp(' ');

%%variabel yang yang dikirim ke pembuatan error
ckirim = tampil;
cterima= ckirim;

%%pembuatan error
hasil = BuatEror('Title','Buat Error');
if lower(hasil)=='batal'
    cterima = ckirim
end;

%%//cterima// data dikirim setelah diubah
%%-----------------
terima=sandi;
for i1=1:256
    if cterima(i1)=='0'
        terima(i1)=0;
    else
        terima(i1)=1;
    end;
end;

%%tabel untuk koreksi
K1=[0 1];
K2=[K1 K1];
K3=[K2 K2];
K4=[K3 K3];
K5=[K4 K4];
K6=[K5 K5];
K7=[K6 K6];
K8=[K7 K7];

%%cek error dari //TERIMA//
digitBenar=0;
digitSalah=0;
```

```
adaError='no';
digitUbah='';
for i1=1:256
    if terima(i1)==K8(i1)
        digitBenar=digitBenar+1;
    else
        digitSalah=digitSalah+1;
    end;
    if sandi(i1)~=terima(i1)
        if adaError=='no'
            digitUbah=['Digit yang diubah : ',num2str(i1-1)];
            adaError='ya';
        else
            digitUbah=[digitUbah,', ',num2str(i1-1)];
        end;
    end;
end;

%% tentukan error berdasarkan nilai Z
zij=abs((digitBenar-digitSalah)/256);
disp('Sandi Kata Terima :');
disp(baris1);
disp(baris2);
disp(char(cterima));   %% kata //TERIMA// dalam bentuk karakter
biner
disp(' ');
disp(['Z ij              : ',num2str(zij)]);
disp(' ');
disp(digitUbah);
disp(' ');

if (zij==0) | (zij==1)
    %% kembalikan terima (256 bit) menjadi (8 bit) menggunakan
tabel H8
    adaError = 'no';
    hasil = -1;
    for i1=1:256
        if terima==H8(i1,1:256)
            %%hasil = index dari tabel H8
            hasil = i1-1;
            break;
        end;
    end;
    chasil=dec2bin(hasil);
    while length(chasil)<8
        chasil=strcat('0',chasil);
    end;

    if hasil==(-1)
        disp('Proses Lookup Gagal');
        adaError='ya';
    else
        disp('Proses Lookup Berhasil');
        disp(' ');
        disp(['Tidak ada error']);
        disp(' ');
        disp(['Kata Terima       ']);
        disp(['----------        ']);
```

```
        disp(['Hasil (BINER)      : ',chasil]);
        disp(['Hasil (DESIMAL)    : ',num2str(hasil)]);
    end;
    disp(' ');
else
    adaError='ya';
end;

%% temukan kataterima jika adaerror dengan n salah terkecil
if adaError=='ya'
    %% hitung d minimal pada H8 ----
    dmin = 300;
    hmin = 300;
    for i2=1:256
        digitBenar=0;
        digitSalah=0;
        L8 = H8(i2,1:256);
        for i1=1:256
            if L8(i1)==terima(i1)
                digitBenar=digitBenar+1;
            else
                digitSalah=digitSalah+1;
            end;
        end;
        if dmin > digitSalah
            dmin = digitSalah;
            hmin = i2-1;
        end;
        %%disp(strcat('error min : ',num2str(abs((digitBenar-
digitSalah)/256))));
    end;
    %% tampilan biner ---
    chmin = dec2bin(hmin);
    while length(chmin)<8
        chmin = strcat('0',chmin);
    end;
    disp('Ada error');
    disp(['E min ( n error)  : ',num2str(dmin)]);
    disp(['Kata Terima       ']);
    disp(['-----------       ']);
    disp(['h min (BINER)     : ',chmin]);
    disp(['h min (DESIMAL)   : ',num2str(hmin)]);
end;

disp(' ');
disp(' ');
ulangiTerus=input('Ulangi Lagi [Y/T] : ','s');

end;

disp(' ');
disp(' ');
%%selesai
return;
```

LAMPIRAN
PROGRAM GUI (Tampilan)

```
function varargout = BuatEror(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @BuatEror_OpeningFcn, ...
                   'gui_OutputFcn',  @BuatEror_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code –

% --- Executes just before BuatEror is made visible.
function BuatEror_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = 'Close';          %% ok!
%% salin ckirim ke text 3 6 9 12
ckirim = char(evalin('base','ckirim'));
set(handles.text3,'string',ckirim(1:70));
set(handles.text6,'string',ckirim(71:140));
set(handles.text9,'string',ckirim(141:210));
set(handles.text12,'string',ckirim(211:256));
%% edit dan pesan --
set(handles.edit1,'string','0');
set(handles.text15,'string','---');
%% system --------
handles.output = hObject;
guidata(hObject, handles);
if(nargin > 3)
   for index = 1:2:(nargin-3),
      if nargin-3==index, break, end
      switch lower(varargin{index})
       case 'title'
        set(hObject, 'Name', varargin{index+1});
       case 'string'
        set(handles.text1, 'String', varargin{index+1});
```

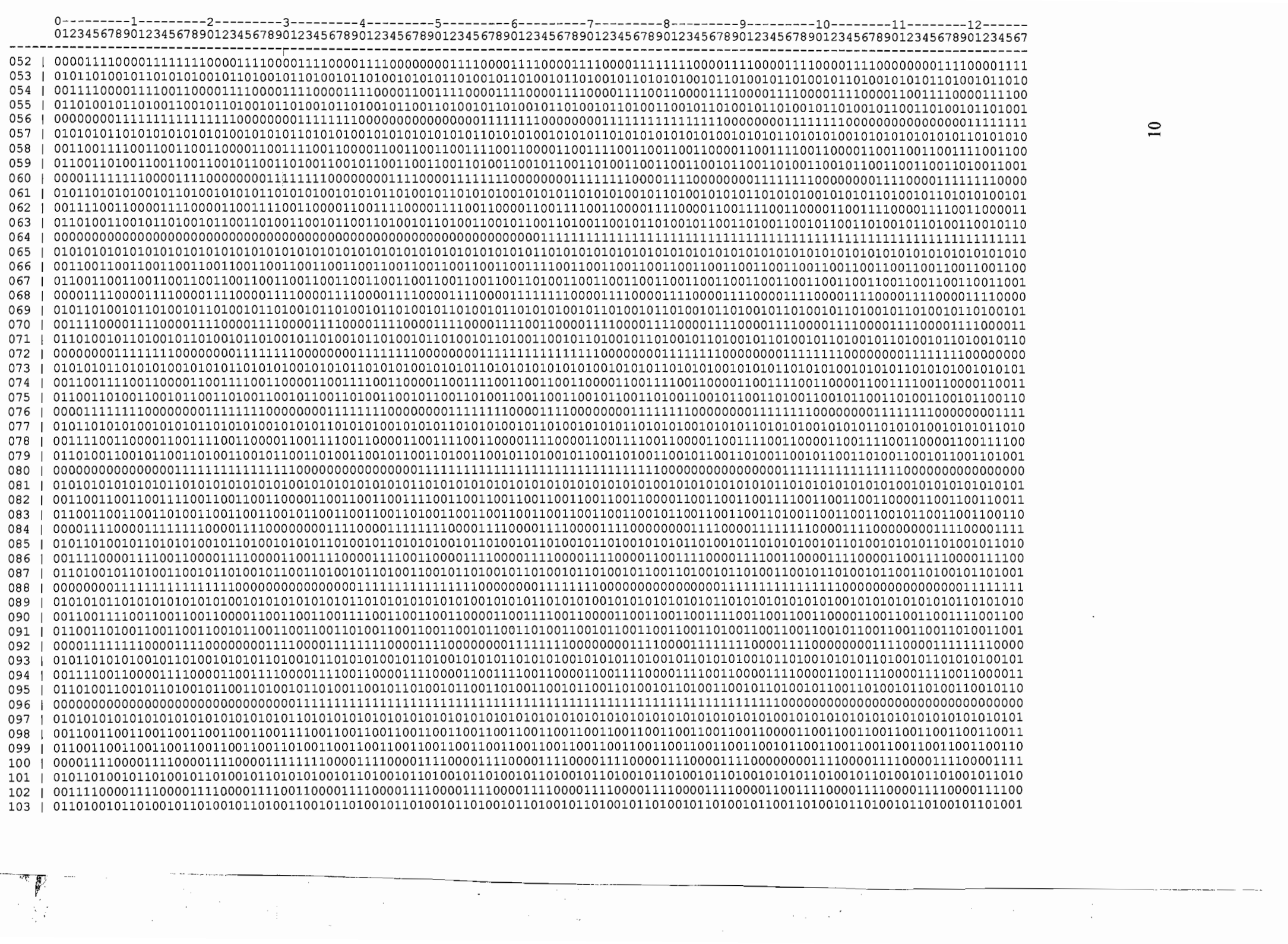
```
    end
  end
end
set(handles.figure1,'WindowStyle','modal')
uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = BuatEror_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
delete(handles.figure1);

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
%% kembalikan terima ke workspace
cterima = strcat(get(handles.text3,'string'),get(handles.text6,'string'),...
        get(handles.text9,'string'),get(handles.text12,'string'));
assignin('base','cterima',cterima);
%% kembalikan proses
handles.output = get(hObject,'String');    %%--Close--
guidata(hObject, handles);
uiresume(handles.figure1);

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
handles.output = get(hObject,'String');    %%--Batalkan--
guidata(hObject, handles);
uiresume(handles.figure1);

% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
if isequal(get(handles.figure1, 'waitstatus'), 'waiting')
  uiresume(handles.figure1);
else
  delete(handles.figure1);
end

% --- Executes on key press over figure1 with no controls selected.
function figure1_KeyPressFcn(hObject, eventdata, handles)
% Check for "enter" or "escape"
if isequal(get(hObject,'CurrentKey'),'escape')
  handles.output = 'Batal';
  guidata(hObject, handles);
  uiresume(handles.figure1);
end
if isequal(get(hObject,'CurrentKey'),'return')
  uiresume(handles.figure1);
end
```

```
% ---
function edit1_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
salahInput = 't';
c1 = get(handles.edit1,'string');
n1 = length(c1);
if n1<1
   salahInput = 'y';
else
   for i1=1:n1
      if (c1(i1)<'0') || (c1(i1)>'9')
         salahInput = 'y';
      end;
   end;
   if salahInput == 't'
      d1 = str2num(c1);
      if (d1<0) || (d1>255)
         salahInput = 'y';
      end;
   end;
end
%%----------%% text 3 6 9 12
cterima = strcat(get(handles.text3,'string'),get(handles.text6,'string'),...
         get(handles.text9,'string'),get(handles.text12,'string'));
if salahInput == 't'
   if cterima(d1+1)=='1'
      cterima(d1+1)='0';
   else
      cterima(d1+1)='1';
   end;
   set(handles.text3,'string',cterima(1:70));
   set(handles.text6,'string',cterima(71:140));
   set(handles.text9,'string',cterima(141:210));
   set(handles.text12,'string',cterima(211:256));
   %% cek digit diubah ---
   ckirim = char(evalin('base','ckirim'));
```

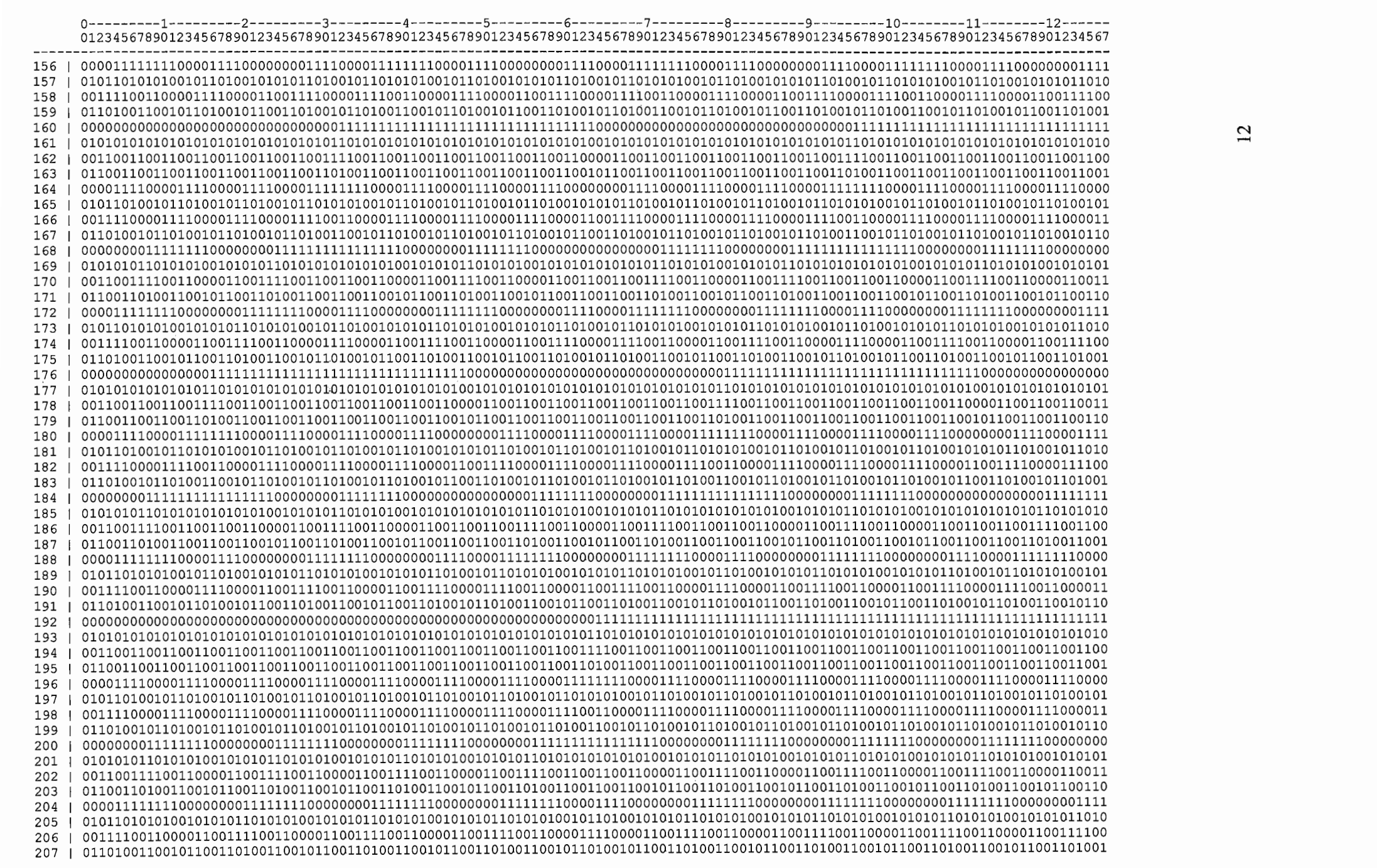
```
  adaError='no';
  digitUbah='';
  for i1=1:256
     if ckirim(i1) ~= cterima(i1)
        if adaError=='no'
           digitUbah=['Digit yang diubah : ',num2str(i1-1)];
           adaError='ya';
        else
           digitUbah=[digitUbah,', ',num2str(i1-1)];
        end;
     end;
  end;
  set(handles.text15,'string',digitUbah);
else
  errordlg('Ketikkan Nilai Bilangan antara 0 s.d. 255','Salah Input','modal');
end
```

LAMPIRAN
Data Set Sandi Orthogonal 8 Bit

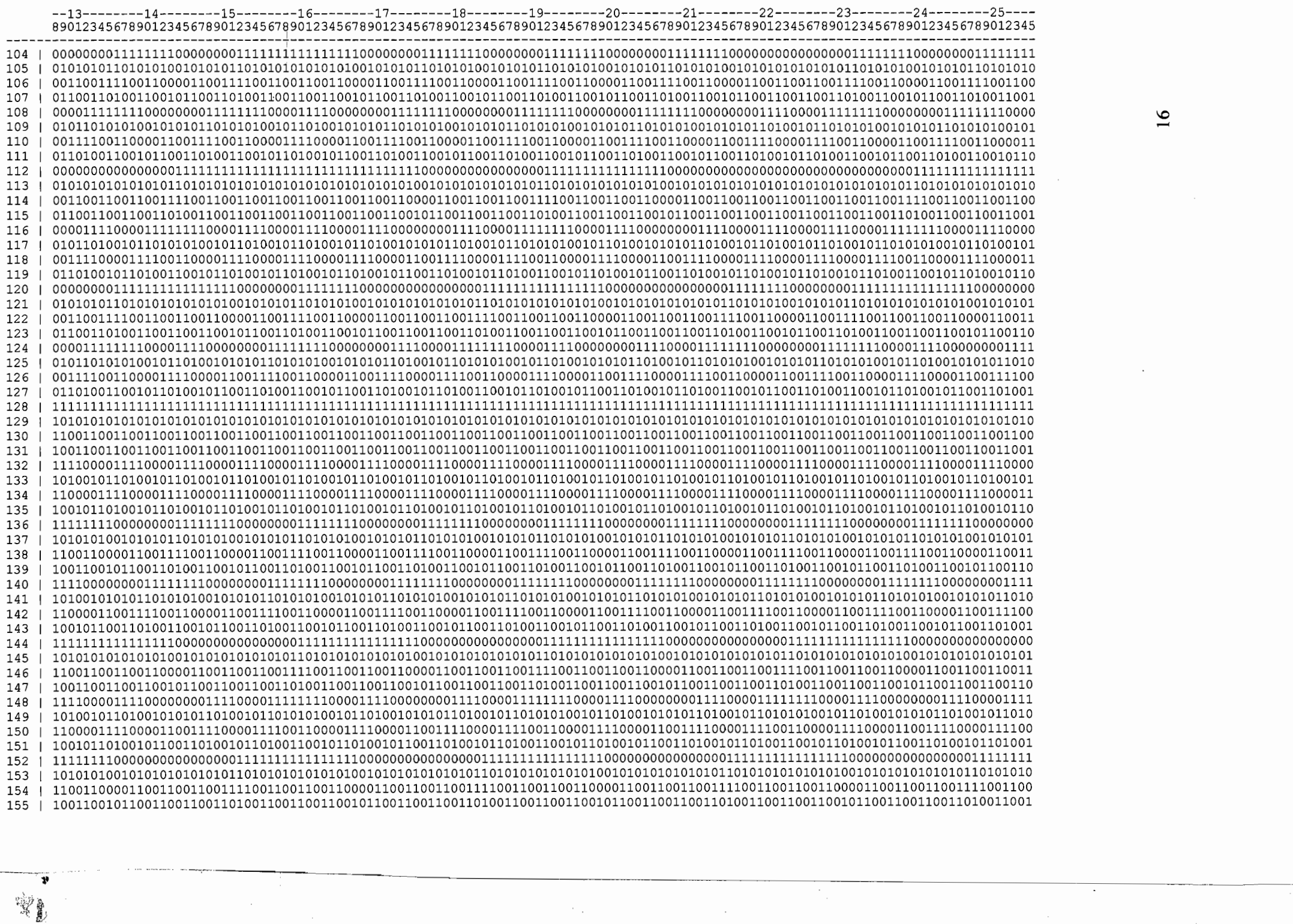
```
      0---------1---------2---------3---------4---------5---------6---------7---------8---------9---------10--------11--------12------
      01234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567
--------------------------------------------------------------------------------------------------------------------------------------
000 | 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
001 | 01010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101
002 | 00110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011
003 | 01100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110
004 | 00001111000011110000111100001111000011110000111100001111000011110000111100001111000011110000111100001111000011110000111100001111
005 | 01011010010110100101101001011010010110100101101001011010010110100101101001011010010110100101101001011010010110100101101001011010
006 | 00111100001111000011110000111100001111000011110000111100001111000011110000111100001111000011110000111100001111000011110000111100
007 | 01101001011010010110100101101001011010010110100101101001011010010110100101101001011010010110100101101001011010010110100101101001
008 | 00000000111111110000000011111111000000001111111100000000111111110000000011111111000000001111111100000000111111110000000011111111
009 | 01010101101010100101010110101010010101011010101001010101101010100101010110101010010101011010101001010101101010100101010110101010
010 | 00110011110011000011001111001100001100111100110000110011110011000011001111001100001100111100110000110011110011000011001111001100
011 | 01100110100110010110011010011001011001101001100101100110100110010110011010011001011001101001100101100110100110010110011010011001
012 | 00001111111100000000111111110000000011111111000000001111111100000000111111110000000011111111000000001111111100000000111111110000
013 | 01011010101001010101101010100101010110101010010101011010101001010101101010100101010110101010010101011010101001010101101010100101
014 | 00111100110000110011110011000011001111001100001100111100110000110011110011000011001111001100001100111100110000110011110011000011
015 | 01101001100101100110100110010110011010011001011001101001100101100110100110010110011010011001011001101001100101100110100110010110
016 | 00000000000000001111111111111111000000000000000011111111111111110000000000000000111111111111111100000000000000001111111111111111
017 | 01010101010101011010101010101010010101010101010110101010101010100101010101010101101010101010101001010101010101011010101010101010
018 | 00110011001100111100110011001100001100110011001111001100110011000011001100110011110011001100110000110011001100111100110011001100
019 | 01100110011001101001100110011001011001100110011010011001100110010110011001100110100110011001100101100110011001101001100110011001
020 | 00001111000011111111000011110000000011110000111111110000111100000000111100001111111100001111000000001111000011111111000011110000
021 | 01011010010110101010010110100101010110100101101010100101101001010101101001011010101001011010010101011010010110101010010110100101
022 | 00111100001111001100001111000011001111000011110011000011110000110011110000111100110000111100001100111100001111001100001111000011
023 | 01101001011010011001011010010110011010010110100110010110100101100110100101101001100101101001011001101001011010011001011010010110
024 | 00000000111111111111111100000000000000001111111111111111000000000000000011111111111111110000000000000000111111111111111100000000
025 | 01010101101010101010101001010101010101011010101010101010010101010101010110101010101010100101010101010101101010101010101001010101
026 | 00110011110011001100110000110011001100111100110011001100001100110011001111001100110011000011001100110011110011001100110000110011
027 | 01100110100110011001100101100110011001101001100110011001011001100110011010011001100110010110011001100110100110011001100101100110
028 | 00001111111100001111000000001111000011111111000011110000000011110000111111110000111100000000111100001111111100001111000000001111
029 | 01011010101001011010010101011010010110101010010110100101010110100101101010100101101001010101101001011010101001011010010101011010
030 | 00111100110000111100001100111100001111001100001111000011001111000011110011000011110000110011110000111100110000111100001100111100
031 | 01101001100101101001011001101001011010011001011010010110011010010110100110010110100101100110100101101001100101101001011001101001
032 | 00000000000000000000000000000000111111111111111111111111111111110000000000000000000000000000000011111111111111111111111111111111
033 | 01010101010101010101010101010101101010101010101010101010101010100101010101010101010101010101010110101010101010101010101010101010
034 | 00110011001100110011001100110011110011001100110011001100110011000011001100110011001100110011001111001100110011001100110011001100
035 | 01100110011001100110011001100110100110011001100110011001100110010110011001100110011001100110011010011001100110011001100110011001
036 | 00001111000011110000111100001111111100001111000011110000111100000000111100001111000011110000111111110000111100001111000011110000
037 | 01011010010110100101101001011010101001011010010110100101101001010101101001011010010110100101101010100101101001011010010110100101
038 | 00111100001111000011110000111100110000111100001111000011110000110011110000111100001111000011110011000011110000111100001111000011
039 | 01101001011010010110100101101001100101101001011010010110100101100110100101101001011010010110100110010110100101101001011010010110
040 | 00000000111111110000000011111111111111110000000011111111000000000000000011111111000000001111111111111111000000001111111100000000
041 | 01010101101010100101010110101010101010100101010110101010010101010101010110101010010101011010101010101010010101011010101001010101
042 | 00110011110011000011001111001100110011000011001111001100001100110011001111001100001100111100110011001100001100111100110000110011
043 | 01100110100110010110011010011001100110010110011010011001011001100110011010011001011001101001100110011001011001101001100101100110
044 | 00001111111100000000111111110000111100000000111111110000000011110000111111110000000011111111000011110000000011111111000000001111
045 | 01011010101001010101101010100101101001010101101010100101010110100101101010100101010110101010010110100101010110101010010101011010
046 | 00111100110000110011110011000011110000110011110011000011001111000011110011000011001111001100001111000011001111001100001100111100
047 | 01101001100101100110100110010110100101100110100110010110011010010110100110010110011010011001011010010110011010011001011001101001
048 | 00000000000000001111111111111111111111111111111100000000000000000000000000000000111111111111111111111111111111110000000000000000
049 | 01010101010101011010101010101010101010101010101001010101010101010101010101010101101010101010101010101010101010100101010101010101
050 | 00110011001100111100110011001100110011001100110000110011001100110011001100110011110011001100110011001100110011000011001100110011
051 | 01100110011001101001100110011001100110011001100101100110011001100110011001100110100110011001100110011001100110010110011001100110
```



```
     0---------1---------2---------3---------4---------5---------6---------7---------8---------9---------10--------11--------12------
     01234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567
-----------------------------------------------------------------------------------------------------------------------------------
104 | 00000000111111100000000111111111111111100000000111111110000000011111111000000000111111110000000000000000011111111000000001111111
105 | 01010101101010100101010110101010101010100101010110101010010101011010101001010101101010100101010101010101101010100101010110101010
106 | 00110011110011000011001111001100110011000011001111001100001100111100110000110011110011000011001100110011110011000011001111001100
107 | 01100110100110010110011010011001100110010110011010011001011001101001100101100110100110010110011001100110100110010110011010011001
108 | 00001111111100000000111111110000111100000000111111110000000011111111000000001111111100000000111100001111111100000000111111110000
109 | 01011010101001010101101010100101101001010101101010100101010110101010010101011010101001010101101001011010101001010101101010100101
110 | 00111100110000110011110011000011110000110011110011000011001111001100001100111100110000110011110000111100110000110011110011000011
111 | 01101001100101100110100110010110100101100110100110010110011010011001011001101001100101100110100101101001100101100110100110010110
112 | 00000000000000011111111111111111111111111111111100000000000000000111111111111111100000000000000000000000000000000011111111111111
113 | 01010101010101011010101010101010101010101010101001010101010101011010101010101010010101010101010101010101010101011010101010101010
114 | 00110011001100111100110011001100110011001100110000110011001100111100110011001100001100110011001100110011001100111100110011001100
115 | 01100110011001101001100110011001100110011001100101100110011001101001100110011001011001100110011001100110011001101001100110011001
116 | 00001111000011111111000011110000111100001111000000001111000011111111000011110000000011110000111100001111000011111111000011110000
117 | 01011010010110101010010110100101101001011010010101011010010110101010010110100101010110100101101001011010010110101010010110100101
118 | 00111100001111001100001111000011110000111100001100111100001111001100001111000011001111000011110000111100001111001100001111000011
119 | 01101001011010011001011010010110100101101001011001101001011010011001011010010110011010010110100101101001011010011001011010010110
120 | 00000000111111111111111100000000111111110000000000000000111111111111111100000000000000001111111100000000111111111111111100000000
121 | 01010101101010101010101001010101101010100101010101010101101010101010101001010101010101011010101001010101101010101010101001010101
122 | 00110011110011001100110000110011110011000011001100110011110011001100110000110011001100111100110000110011110011001100110000110011
123 | 01100110100110011001100101100110100110010110011001100110100110011001100101100110011001101001100101100110100110011001100101100110
124 | 00001111111100001111000000001111111100000000111100001111111100001111000000001111000011111111000000001111111100001111000000001111
125 | 01011010101001011010010101011010101001010101101001011010101001011010010101011010010110101010010101011010101001011010010101011010
126 | 00111100110000111100001100111100110000110011110000111100110000111100001100111100001111001100001100111100110000111100001100111100
127 | 01101001100101101001011001101001100101100110100101101001100101101001011001101001011010011001011001101001100101101001011001101001
128 | 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
129 | 01010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101
130 | 00110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011
131 | 01100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110
132 | 00001111000011110000111100001111000011110000111100001111000011110000111100001111000011110000111100001111000011110000111100001111
133 | 01011010010110100101101001011010010110100101101001011010010110100101101001011010010110100101101001011010010110100101101001011010
134 | 00111100001111000011110000111100001111000011110000111100001111000011110000111100001111000011110000111100001111000011110000111100
135 | 01101001011010010110100101101001011010010110100101101001011010010110100101101001011010010110100101101001011010010110100101101001
136 | 00000000111111110000000011111111000000001111111100000000111111110000000011111111000000001111111100000000111111110000000011111111
137 | 01010101101010100101010110101010010101011010101001010101101010100101010110101010010101011010101001010101101010100101010110101010
138 | 00110011110011000011001111001100001100111100110000110011110011000011001111001100001100111100110000110011110011000011001111001100
139 | 01100110100110010110011010011001011001101001100101100110100110010110011010011001011001101001100101100110100110010110011010011001
140 | 00001111111100000000111111110000000011111111000000001111111100000000111111110000000011111111000000001111111100000000111111110000
141 | 01011010101001010101101010100101010110101010010101011010101001010101101010100101010110101010010101011010101001010101101010100101
142 | 00111100110000110011110011000011001111001100001100111100110000110011110011000011001111001100001100111100110000110011110011000011
143 | 01101001100101100110100110010110011010011001011001101001100101100110100110010110011010011001011001101001100101100110100110010110
144 | 00000000000000001111111111111111000000000000000011111111111111110000000000000000111111111111111100000000000000001111111111111111
145 | 01010101010101011010101010101010010101010101010110101010101010100101010101010101101010101010101001010101010101011010101010101010
146 | 00110011001100111100110011001100001100110011001111001100110011000011001100110011110011001100110000110011001100111100110011001100
147 | 01100110011001101001100110011001011001100110011010011001100110010110011001100110100110011001100101100110011001101001100110011001
148 | 00001111000011111111000011110000000011110000111111110000111100000000111100001111111100001111000000001111000011111111000011110000
149 | 01011010010110101010010110100101010110100101101010100101101001010101101001011010101001011010010101011010010110101010010110100101
150 | 00111100001111001100001111000011001111000011110011000011110000110011110000111100110000111100001100111100001111001100001111000011
151 | 01101001011010011001011010010110011010010110100110010110100101100110100101101001100101101001011001101001011010011001011010010110
152 | 00000000111111111111111100000000000000001111111111111111000000000000000011111111111111110000000000000000111111111111111100000000
153 | 01010101101010101010101001010101010101011010101010101010010101010101010110101010101010100101010101010101101010101010101001010101
154 | 00110011110011001100110000110011001100111100110011001100001100110011001111001100110011000011001100110011110011001100110000110011
155 | 01100110100110011001100101100110011001101001100110011001011001100110011010011001100110010110011001100110100110011001100101100110
```




SANATA DHARMA
UNIVERSITAS
YOGYAKARTA


--13--------14--------15--------16--------17--------18--------19--------20--------21--------22--------23--------24--------25----
8901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155

```
      --13--------14--------15--------16--------17--------18--------19--------20--------21--------22--------23--------24--------25----
      89012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345
--------------------------------------------------------------------------------------------------------------------------------------
156 | 11110000000011110000111111110000111100000000111100001111111100001111000000001111000011111111000011110000000011110000111111110000
157 | 10100101010110100101101010100101101001010101101001011010101001011010010101011010010110101010010110100101010110100101101010100101
158 | 11000011001111000011110011000011110000110011110000111100110000111100001100111100001111001100001111000011001111000011110011000011
159 | 10010110011010010110100110010110100101100110100101101001100101101001011001101001011010011001011010010110011010010110100110010110
160 | 11111111111111111111111111111111000000000000000000000000000000001111111111111111111111111111111100000000000000000000000000000000
161 | 10101010101010101010101010101010010101010101010101010101010101011010101010101010101010101010101001010101010101010101010101010101
162 | 11001100110011001100110011001100001100110011001100110011001100111100110011001100110011001100110000110011001100110011001100110011
163 | 10011001100110011001100110011001011001100110011001100110011001101001100110011001100110011001100101100110011001100110011001100110
164 | 11110000111100001111000011110000000011110000111100001111000011111111000011110000111100001111000000001111000011110000111100001111
165 | 10100101101001011010010110100101010110100101101001011010010110101010010110100101101001011010010101011010010110100101101001011010
166 | 11000011110000111100001111000011001111000011110000111100001111001100001111000011110000111100001100111100001111000011110000111100
167 | 10010110100101101001011010010110011010010110100101101001011010011001011010010110100101101001011001101001011010010110100101101001
168 | 11111111000000001111111100000000000000001111111100000000111111111111111100000000111111110000000000000000111111110000000011111111
169 | 10101010010101011010101001010101010101011010101001010101101010101010101001010101101010100101010101010101101010100101010110101010
170 | 11001100001100111100110000110011001100111100110000110011110011001100110000110011110011000011001100110011110011000011001111001100
171 | 10011001011001101001100101100110011001101001100101100110100110011001100101100110100110010110011001100110100110010110011010011001
172 | 11110000000011111111000000001111000011111111000000001111111100001111000000001111111100000000111100001111111100000000111111110000
173 | 10100101010110101010010101011010010110101010010101011010101001011010010101011010101001010101101001011010101001010101101010100101
174 | 11000011001111001100001100111100001111001100001100111100110000111100001100111100110000110011110000111100110000110011110011000011
175 | 10010110011010011001011001101001011010011001011001101001100101101001011001101001100101100110100101101001100101100110100110010110
176 | 11111111111111110000000000000000000000000000000011111111111111111111111111111111000000000000000000000000000000001111111111111111
177 | 10101010101010100101010101010101010101010101010110101010101010101010101010101010010101010101010101010101010101011010101010101010
178 | 11001100110011000011001100110011001100110011001111001100110011001100110011001100001100110011001100110011001100111100110011001100
179 | 10011001100110010110011001100110011001100110011010011001100110011001100110011001011001100110011001100110011001101001100110011001
180 | 11110000111100000000111100001111000011110000111111110000111100001111000011110000000011110000111100001111000011111111000011110000
181 | 10100101101001010101101001011010010110100101101010100101101001011010010110100101010110100101101001011010010110101010010110100101
182 | 11000011110000110011110000111100001111000011110011000011110000111100001111000011001111000011110000111100001111001100001111000011
183 | 10010110100101100110100101101001011010010110100110010110100101101001011010010110011010010110100101101001011010011001011010010110
184 | 11111111000000000000000011111111000000001111111111111111000000001111111100000000000000001111111100000000111111111111111100000000
185 | 10101010010101010101010110101010010101011010101010101010010101011010101001010101010101011010101001010101101010101010101001010101
186 | 11001100001100110011001111001100001100111100110011001100001100111100110000110011001100111100110000110011110011001100110000110011
187 | 10011001011001100110011010011001011001101001100110011001011001101001100101100110011001101001100101100110100110011001100101100110
188 | 11110000000011110000111111110000000011111111000011110000000011111111000000001111000011111111000000001111111100001111000000001111
189 | 10100101010110100101101010100101010110101010010110100101010110101010010101011010010110101010010101011010101001011010010101011010
190 | 11000011001111000011110011000011001111001100001111000011001111001100001100111100001111001100001100111100110000111100001100111100
191 | 10010110011010010110100110010110011010011001011010010110011010011001011001101001011010011001011001101001100101101001011001101001
192 | 11111111111111111111111111111111111111111111111111111111111111110000000000000000000000000000000000000000000000000000000000000000
193 | 10101010101010101010101010101010101010101010101010101010101010100101010101010101010101010101010101010101010101010101010101010101
194 | 11001100110011001100110011001100110011001100110011001100110011000011001100110011001100110011001100110011001100110011001100110011
195 | 10011001100110011001100110011001100110011001100110011001100110010110011001100110011001100110011001100110011001100110011001100110
196 | 11110000111100001111000011110000111100001111000011110000111100000000111100001111000011110000111100001111000011110000111100001111
197 | 10100101101001011010010110100101101001011010010110100101101001010101101001011010010110100101101001011010010110100101101001011010
198 | 11000011110000111100001111000011110000111100001111000011110000110011110000111100001111000011110000111100001111000011110000111100
199 | 10010110100101101001011010010110100101101001011010010110100101100110100101101001011010010110100101101001011010010110100101101001
200 | 11111111000000001111111100000000111111110000000011111111000000000000000011111111000000001111111100000000111111110000000011111111
201 | 10101010010101011010101001010101101010100101010110101010010101010101010110101010010101011010101001010101101010100101010110101010
202 | 11001100001100111100110000110011110011000011001111001100001100110011001111001100001100111100110000110011110011000011001111001100
203 | 10011001011001101001100101100110100110010110011010011001011001100110011010011001011001101001100101100110100110010110011010011001
204 | 11110000000011111111000000001111111100000000111111110000000011110000111111110000000011111111000000001111111100000000111111110000
205 | 10100101010110101010010101011010101001010101101010100101010110100101101010100101010110101010010101011010101001010101101010100101
206 | 11000011001111001100001100111100110000110011110011000011001111000011110011000011001111001100001100111100110000110011110011000011
207 | 10010110011010011001011001101001100101100110100110010110011010010110100110010110011010011001011001101001100101100110100110010110
```


--13--------14--------15--------16--------17--------18--------19--------20--------21--------22--------23--------24--------25----
8901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255